EXHIBIT 1

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

INTERNATIONAL BUSINESS MACHINES CORPORATION, Plaintiff, v. GROUPON, INC. Defendant.	C.A. No. 1:16-cv-00122-LPS

DEFENDANT GROUPON, INC.'S NOTICE REGARDING AUTHENTICITY OBJECTIONS TO TRIAL EXHIBIT LIST OF INTERNATIONAL BUSINESS MACHINES CORPORATION

Pursuant to the Court's June 27, 2018 order granting the June 20, 2018 joint stipulation of the parties (D.I. 323), Defendant Groupon, Inc. ("Groupon") hereby provides Plaintiff International Business Machines Corporation ("IBM") with notice of the authenticity objections to exhibits on IBM's trial exhibit list, which Groupon intends to maintain. Groupon reserves the right to amend, modify, or supplement this notice, consistent with the Court's orders.

Groupon will maintain authenticity objections to the following documents for the reasons set forth below:

- PX-0578-585, PX-0807-827, PX-0828-872, PX-0891-911: These exhibits include handwritten notes, drafts, printouts, overviews, specifications, and source code that purport to relate to systems related to the patents-in-suit; however, there is no evidence in the record authenticating these exhibits and IBM has not established, and the documents do not show, facts establishing the authenticity of these exhibits, including the

source, date, author or circumstances of their creation.

- PX-0604, PX-0606, PX-0612, PX-0615, PX-0619, PX-0755, PX-0759, PX-1075, PX-1095: These exhibits appear to be webpages and third party documents for which IBM has not established, and the documents do not show, facts establishing the authenticity of these exhibits, including the source or circumstances of their creation or publication.
- PX-0701-074, PX-0706: These exhibits appear to be portions of file histories for the asserted patents; however, they are not certified and therefore IBM has not established the source or authenticity of these documents. The certified file histories are already exhibits on IBM's exhibit list.
- PX-0122, PX-0123: These exhibits purport to be Akamai source code and related documentation; however, there is no evidence in the record authenticating these exhibits and IBM has not established, and the documents do not show, facts establishing the authenticity of these exhibits, including the source, date, author or circumstances of their creation.
- PX-1556-1563: These exhibits purport to be source code file comparisons that were not produced by any party or in response to any subpoena in this action. IBM has not established how these documents were created and by whom and under what circumstances.

//

//

Respectfully submitted,

ASHBY & GEDDES

By: */s/ Saina S. Shamilov*

Saina S. Shamilov
John G. Day (#2403)
Andrew C. Mayo (#5207)
500 Delaware Avenue, 8th Floor
P.O. Box 1150
Wilmington, DE 19899
(302) 654-1888
jday@ashby-geddes.com
amayo@ashby-geddes.com

Attorneys for Defendant
GROUPON, INC.

Of counsel:

J. David Hadden
Saina S. Shamilov
Phillip J. Haack
Sapna Mehta
Jessica Kaempf
Athul Acharya
Jessica Benzler
FENWICK & WEST LLP
Silicon Valley Center
801 California Street
Mountain View, CA 94041
Telephone: 650.988.8500
Facsimile: 650.938.5200

Dated: July 3, 2018

CERTIFICATE OF SERVICE

I hereby certify that on this 3rd day of July 2018, a true and correct copy of the foregoing document was served on each party through their counsel of record via email.

John M. Desmarais
Karim Oussayef
Robert C. Harrits
Laurie N. Stempler
Jon T. Hohenthaner
DESMARAIS LLP
230 Park Avenue
New York, NY 10169
Tel: (212) 351-3400
IBMGrouponService@desmaraisllp.com
jdesmarais@desmaraisllp.com
koussayef@desmaraisllp.com
rharrits@desmaraisllp.com
lstempler@desmaraisllp.com
jhohenthaner@desmaraislp.com

Of Counsel for Plaintiff International Business Machines Corporation

David E. Moore (#3983)
Bindu A. Palapura (#5370)
Stephanie E. O'Byrne (#4446)
POTTER ANDERSON & CORROON LLP
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19801
Tel: (302) 984-6000
dmoore@potteranderson.com
bpalapura@potteranderson.com
Counsel for Plaintiff International Business Machines Corporation.

/s/ *Jessica Benzler*
Jessica Benzler
jbenzler@fenwick.com
FENWICK & WEST LLP
Silicon Valley Center
801 California Street
Mountain View, California 94041
Telephone: (650) 988-8500
Facsimile: (650) 938-5200

Attorneys for Defendant
GROUPON, INC.

EXHIBIT 2

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

INTERNATIONAL BUSINESS MACHINES CORPORATION, Plaintiff, v. GROUPON, INC. Defendant.	C.A. No. 1:16-cv-00122-LPS

ELECTION OF PRIOR ART FOR INVALIDITY DEFENSES OF DEFENDANT GROUPON, INC.

Pursuant to the Court's June 15, 2018 order (D.I. 314), the Court's June 18, 2018 order granting the joint stipulation of the parties (D.I. 315), and the parties' June 20, 2018 joint stipulation (D.I. 323), Groupon hereby discloses the prior art per asserted patent that it may contend renders the asserted claims invalid at trial. Nothing herein limits, or intends to limit, the invalidity theories Groupon may advance at trial, evidence upon which Groupon may rely to establish the state of the art of the asserted patents or any authentication or background evidence.

I. U.S. PATENT NO. 5,796,967

	Prior Art	Trial Exhibit Number(s)
1	"Andrew: A Distributed Personal Computing Environment," James H. Morris	DX-0355
2	Xerox Star	DX-0356, DX-0357
3	HyperCard	DX-0351, DX-0352, DX-0354, DX-0394, DX-0643

4	"Caching Hints in Distributed Systems," Douglas B. Terry, Journal IEEE Transactions on Software Engineering, Vol. 13 (January 1987)	DX-0358

II. U.S. PATENT NO. 7,072,849

	Prior Art Reference	Trial Exhibit Number(s)
5	"VIDEOTEX/TELETEXT Principles and Practices," Antoine F. Alber (1985)	DX-0366
6	U.S. Patent No. 4,575,579 to Simon (Mar. 11, 1986)	DX-0365
7	"Design and Implementation of An Electronic Special Interest Maga-zine" Gitta B. Salomon (September 1986)	DX-0348
8	IBM's Trintex System (1987)	DX-0158, DX-0359, DX-0360, DX-0361, DX-0362, DX-0363, DX-0364

III. U.S. PATENT NO. 5,961,601

	Prior Art Reference	Trial Exhibit Number(s)
9	"HTML & CGI UNLEASHED" and accompanying CD including source code, John December	DX-0058, DX-0167, DX-0373
10	"Spinning the Web: a guide to serving information on the World Wide Web," Yuval Fisher (February 23, 1996)	DX-0374, DX-0442
11	U.S. Patent No. 6,016,484 to Williams (Jan. 18, 2000)	DX-0377
12	Amazon.com system	DX-0202, DX-0375, DX-0376, PX-1544 – PX-1563

IV. U.S. PATENT NO. 7,631,346

	Prior Art Reference	Trial Exhibit Number(s)
13	Liberty Alliance System Specifications	DX-0380, DX-0381, DX-0382, DX-0383, DX-0645, DX-0646, DX-0647
14	Japanese Patent Application Publication No. 2004-302907 to Sunada (Oct. 28, 2004)	DX-0378, DX-0525
15	U.S. Patent No. 7,680,819 to Mellmer (Mar. 16, 2010)	DX-0379
16	U.S. Patent No. 7,137,006 to Grandcolas (Nov. 14, 2006)	DX-0384

Respectfully submitted,

ASHBY & GEDDES

By: */s/ Saina S. Shamilov*
Saina S. Shamilov
John G. Day (#2403)
Andrew C. Mayo (#5207)
500 Delaware Avenue, 8th Floor
P.O. Box 1150
Wilmington, DE 19899
(302) 654-1888
jday@ashby-geddes.com
amayo@ashby-geddes.com

Attorneys for Defendant
GROUPON, INC.

Of counsel:

J. David Hadden
Saina S. Shamilov
Phillip J. Haack
Sapna Mehta
Jessica Kaempf
Athul Acharya
Jessica Benzler
FENWICK & WEST LLP
Silicon Valley Center
801 California Street
Mountain View, CA 94041
Telephone: 650.988.8500
Facsimile: 650.938.5200

Dated: June 26, 2018

CERTIFICATE OF SERVICE

I hereby certify that on this 26th day of June 2018, a true and correct copy of the foregoing document was served on each party through their counsel of record via email.

John M. Desmarais
Karim Oussayef
Robert C. Harrits
Laurie N. Stempler
Brian D. Matty
Michael J.X. Matulewicz-Crowley
DESMARAIS LLP
230 Park Avenue
New York, NY 10169
Tel: (212) 351-3400
IBMGrouponService@desmaraisllp.com
jdesmarais@desmaraisllp.com
koussayef@desmaraisllp.com
rharrits@desmaraisllp.com
lstempler@desmaraisllp.com
bmatty@desmaraisllp.com
mmatulewicz-crowley@dllp.com

Of Counsel for Plaintiff International Business Machines Corporation

David E. Moore (#3983)
Bindu A. Palapura (#5370)
Stephanie E. O'Byrne (#4446)
POTTER ANDERSON & CORROON LLP
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19801
Tel: (302) 984-6000
dmoore@potteranderson.com
bpalapura@potteranderson.com
Counsel for Plaintiff International Business Machines Corporation.

/s/ *Jessica Benzler*
Jessica Benzler
jbenzler@fenwick.com
FENWICK & WEST LLP
Silicon Valley Center
801 California Street
Mountain View, California 94041
Telephone: (650) 988-8500
Facsimile: (650) 938-5200

Attorneys for Defendant
GROUPON, INC.

EXHIBIT 3

Liberty ID-FF Architecture Overview

Version: 1.2-errata-v1.0

Editors:
Thomas Wason, IEEE-ISTO

Contributors:
Scott Cantor, Internet2, The Ohio State University
Jeff Hodges, Sun Microsystems, Inc.
John Kemp, IEEE-ISTO
Peter Thompson, IEEE-ISTO

Abstract:

This is a non-normative document describing the basic structure and operation of the Liberty Alliance architecture. Examples are provided to illustrate the operation of systems using the architecture. It is intended that this document provide a general introduction to the Liberty ID-FF architecture.

Filename: draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf

Notice

Liberty Alliance Project
Licensing Administrator
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08855-1331, USA
info@projectliberty.org

Contents

1. Introduction

The Internet is now a prime vehicle for business, community, and personal interactions. The notion of *identity* is the crucial component of this vehicle. Today, one's identity on the Internet is fragmented across various identity providers, employers, Internal portals, various communities, and business services. This fragmentation yields isolated, high-friction, one-to-one customer-to-business relationships and experiences.

Federated network identity is the key to reducing this friction and realizing new business taxonomies and opportunities, coupled with new economies of scale. In this new world of federated commerce, a user's online identity, personal profile, personalized online configurations, buying habits and history, and shopping preferences will be administered by the user and securely shared with the organizations of the user's choosing. A federated network identity model will ensure that critical private information is used by appropriate parties.

The path to realizing a rich, fertile federated identity infrastructure can be taken in phases. The natural first phase is the establishment of a standardized, multivendor, Web-based single sign-on with simple federated identities based on today's commonly deployed technologies. This document presents an overview of the *Liberty Identity Federation Framework (ID-FF)*, which offers a viable approach for implementing such a single sign-on with federated identities. This overview first summarizes federated network identity, describes two key Liberty ID-FF user experience scenarios, summarizes the ID-FF engineering requirements and security framework, and then provides a discussion of the Liberty ID-FF architecture.

1.1. About This Document

This document is *non-normative*. However, it provides implementers and deployers guidance in the form of policy/security and technical notes. Further details of the Liberty ID-FF architecture are given in several normative technical documents associated with this overview, specifically [LibertyAuthnContext], [LibertyBindProf], [LibertyImplGuide], and [LibertyProtSchema]. Note: The more global term *Principal* is used for *user* in Liberty's technical documents. Definitions for Liberty-specific terms can be found in the [LibertyGlossary]. Also, many abbreviations are used in this document without immediate definition because the authors believe these abbreviations are widely known, for example, HTTP and SSL. However, the definitions of these abbreviations can also be found in [LibertyGlossary]. Note: Phrases and numbers in brackets [ ] refer to other documents; details of these references can be found in References (at the end of this document). As this document is non-normative it does not use terminology "MUST", "MAY", "SHOULD" in a manner consistent with RFC-2119 (see [RFC2119]).

1.2. What is the Liberty Alliance?

The Liberty Alliance Project represents a broad spectrum of industries united to drive a new level of trust, commerce, and communications on the Internet.

1.2.1. The Liberty Vision

The members of the Liberty Alliance envision a networked world across which individuals and businesses can engage in virtually any transaction without compromising the privacy and security of vital identity information.

1.2.2. The Liberty Mission

To accomplish its vision, the Liberty Alliance will establish open technical specifications that support a broad range of network identity-based interactions and provide businesses with

- A basis for new revenue opportunities that economically leverage their relationships with consumers and business partners and
- A framework within which the businesses can provide consumers with choice, convenience, and control when using any device connected to the Internet.

1.3. What is Network Identity?

When users interact with services on the Internet, they often tailor the services in some way for their personal use. For example, a user may establish an account with a username and password and/or set some preferences for what information the user wants displayed and how the user wants it displayed. The network identity of each user is the overall global set of these attributes constituting the various accounts (see Figure 1).

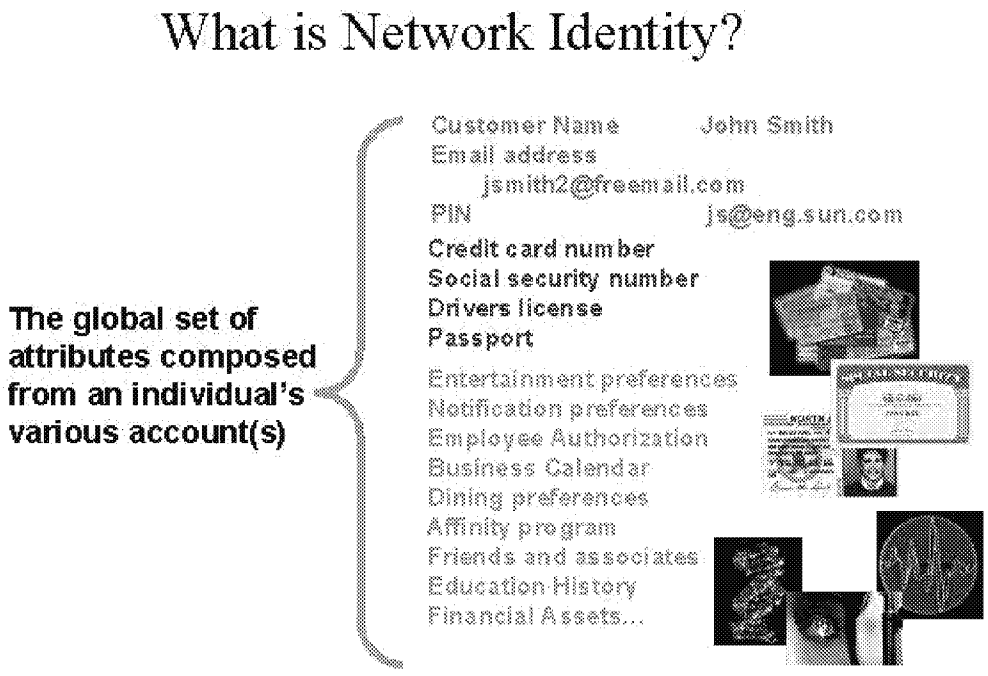


Figure 1. A network identity is the global set of attributes composed from a user's account(s).

Today, users' accounts are scattered across isolated Internet sites. Thus the notion that a user could have a cohesive, tangible network identity is not realized.

1.3.1. The Liberty Objectives

The key objectives of the Liberty Alliance are to:

- Enable consumers to protect the privacy and security of their network identity information
- Enable businesses to maintain and manage their customer relationships without third-party participation
- Provide an open single sign-on standard that includes decentralized authentication and authorization from multiple providers
- Create a network identity infrastructure that supports all current and emerging network access devices

These capabilities can be achieved when, first, businesses affiliate together into *circles of trust* based on Liberty-enabled technology and on operational agreements that define *trust relationships* between the businesses and, second, users federate the otherwise isolated accounts they have with these businesses (known as their *local identities*). In other words, a circle of trust is a federation of service providers and identity providers that have business relationships based on Liberty architecture and operational agreements and with whom users can transact business in a secure and apparently seamless environment. See Figure 2. Note: Operational agreement definitions are out of the scope of the Liberty Version 1.2 specifications.

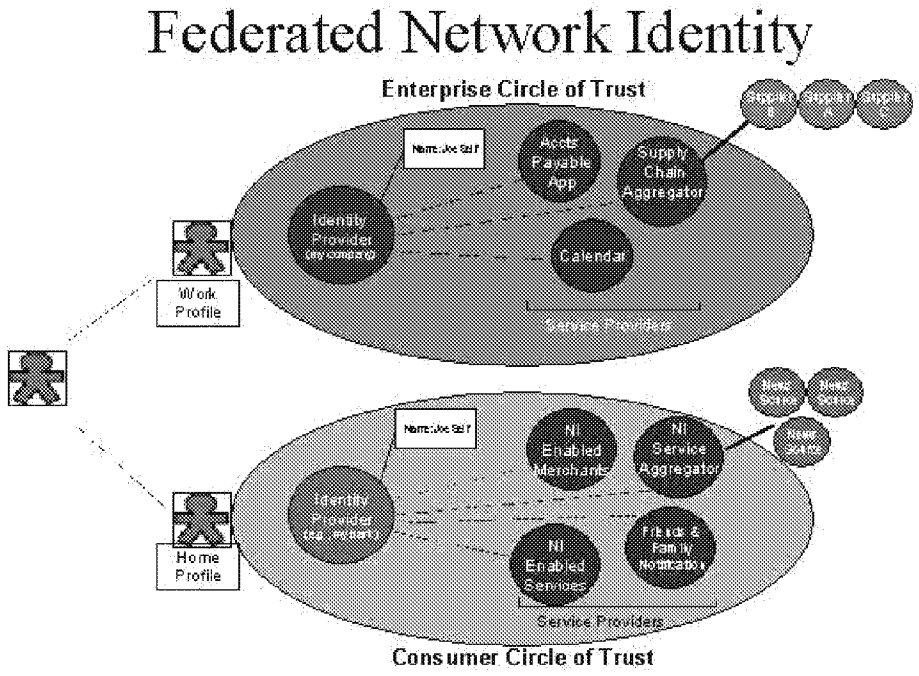


Figure 2. Federated network identity and circles of trust

From a Liberty perspective, the salient actors in Figure 2 are the user, service providers, and identity providers.

Service providers are organizations offering Web-based services to users. This broad category includes practically any organization on the Web today, for example, Internet portals, retailers, transportation providers, financial institutions, entertainment companies, not-for-profit organizations, governmental agencies, etc.

Identity providers are service providers offering business incentives so that other service providers affiliate with them. Establishing such relationships creates the circles of trust shown in Figure 2. For example, in the enterprise circle of trust, the identity provider is a company leveraging employee network identities across the enterprise. Another example is the consumer circle of trust, where the user's bank has established business relationships with various other service providers allowing the user to wield his/her bank-based network identity with them. Note: A single organization may be both an identity provider and a service provider, either generally or for a given interaction.

These scenarios are enabled by service providers and identity providers deploying Liberty-enabled products in their infrastructure, but do not require users to use anything other than today's common Web browser.

2. Liberty ID-FF User Experience Examples

This section provides two simple, plausible examples of the Liberty ID-FF user experience, from the perspective of the user, to set the overall context for delving into technical details of the Liberty architecture in Section 4. As such, actual technical details are hidden or simplified.

Note: the user experience examples presented in this section are non-normative and are presented for illustrative purposes only.

These user experience examples are based upon the following set of actors:

Joe Self	A user of Web-based online services.
Airline.inc	An airline maintaining an affinity group of partners. Airline.inc is an identity provider.
CarRental.inc	A car rental company that is a member of the airline's affinity group. CarRental.inc is a service provider.

The Liberty ID-FF user experience has two main facets:

- Identity federation
- Single sign-on

Identity federation is based upon linking users' otherwise distinct service provider and identity provider accounts. This account linkage, or *identity federation*, in turn underlies and enables the other facets of the Liberty ID-FF user experience.

OVERALL POLICY/SECURITY NOTE:

Identity federation must be predicated upon prior agreement between the identity and service providers. It should be additionally predicated upon providing notice to the user, obtaining the user's consent, and recording both the notice and consent in an auditable fashion. Providing an auditable record of notice and consent will enable both users and providers to confirm that notice and consent were provided and to document that the consent is bound to a particular interaction. Such documentation will increase consumer trust in online services. Implementors and deployers of Liberty-enabled technology should ensure that notice and user consent are auditably recorded in Liberty-enabled interactions with users, as appropriate.

Single sign-on enables users to sign on once with a member of a federated group of identity and service providers (or, from a provider's point of view, with a member of a circle of trust) and subsequently use various Websites among the group without signing on again.

2.1. Example of Identity Federation User Experience

The identity federation facet of the Liberty ID-FF user experience typically begins when Joe Self logs in to Airline.inc's Website, a Liberty-enabled identity provider, as illustrated in Figure 3.

Note:

Even though Joe Self is unaware of it, behind the scenes the identity provider is using Joe Self's credentials-his username and password in this case-to authenticate his identity. If successful, Joe Self is considered *authenticated*.

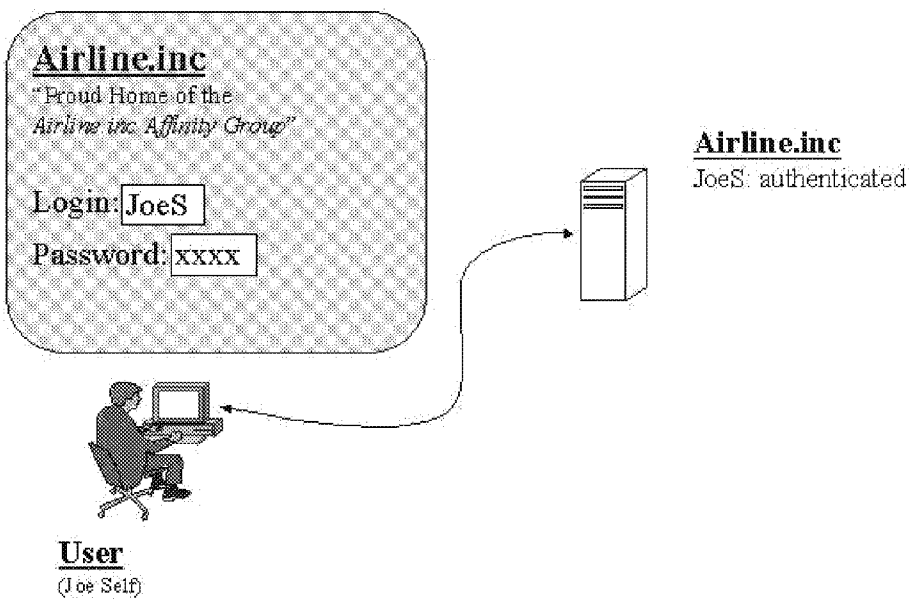
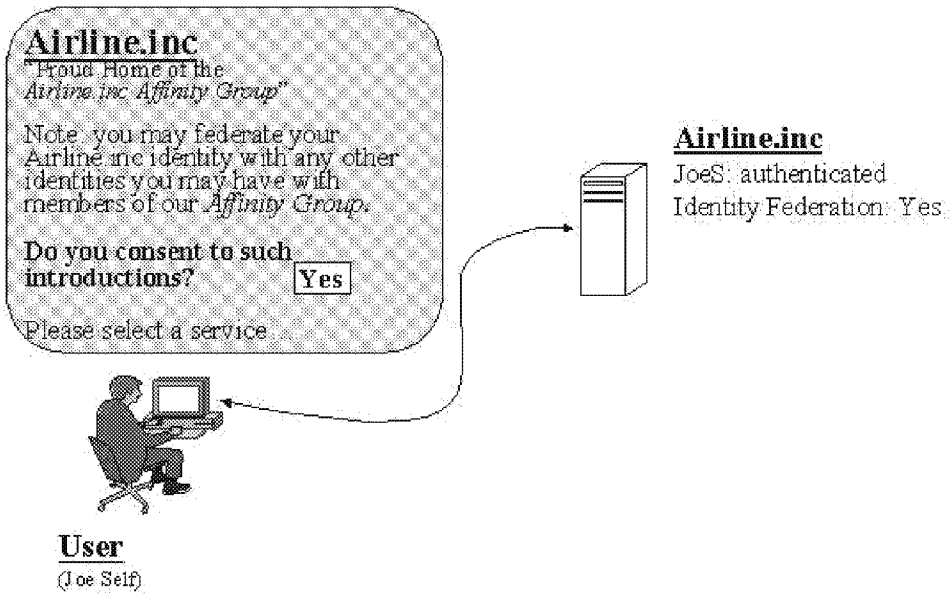


Figure 3. User logs in at a Liberty-enabled Website.

Airline.inc. (as would any other identity provider that has created a circle of trust among its affinity group) will notify its eligible users of the possibility of federating their local identities among the members of the affinity group and will solicit permission to facilitate the introduction of the user to the members of the affinity group. See Figure 4.



155

156 **Figure 4. User is notified of eligibility for identity federation and elects to allow introductions.**

157 **POLICY/SECURITY NOTE:**

158 Figure 4 illustrates the user's consent to being introduced to members of the affinity group. Such an
159 introduction is the means by which a service provider may discover which identity providers in the circle
160 of trust have authenticated the user.

161 In Figure 4 the user is not consenting to federating his identity with any service providers. Soliciting consent
162 to identity federation is a separate step, as illustrated in Figure 5.

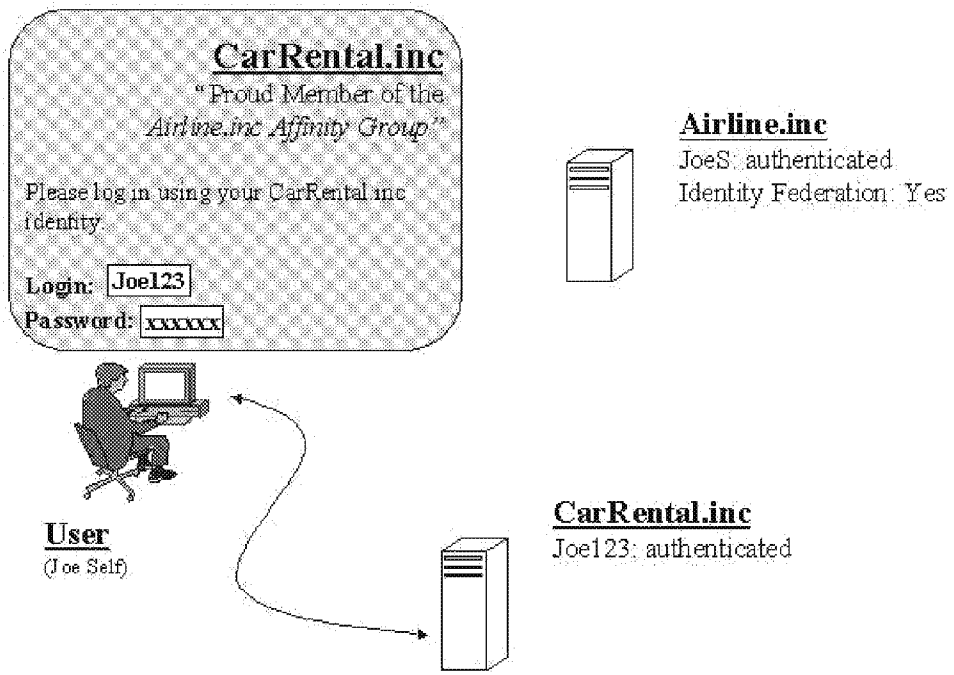
163 Introduction of the user to the affinity group members may be achieved via the Identity Provider Introduction
164 Profile (as detailed in [LibertyBindProf]), or via other unspecified means, such as when the user agent is a
165 Liberty-enabled client or proxy (LEC/P).

166 At some later point in time, typically minutes to a few hours, Joe Self may visit the Website of an affinity group
167 member, for example, CarRental, Inc., whose site is CarRental.inc. Indeed, Joe Self may have followed an explicit
168 link from the original Airline.inc Website to the CarRental.inc Website. In either case, CarRental.inc (a Liberty-
169 enabled service provider) is able to discern that Joe Self recently interacted with the Airline.inc Website, because Joe
170 Self elected to allow introductions.

171 **TECHNICAL NOTE:**

172 The actual means used to perform the introduction is an implementation and deployment decision. One
173 possible means, the Identity Provider Introduction profile, is specified in [LibertyBindProf]. Note that the
174 user may or may not need to log in in order to facilitate introduction - this depends on the specific introduction
175 technique used.

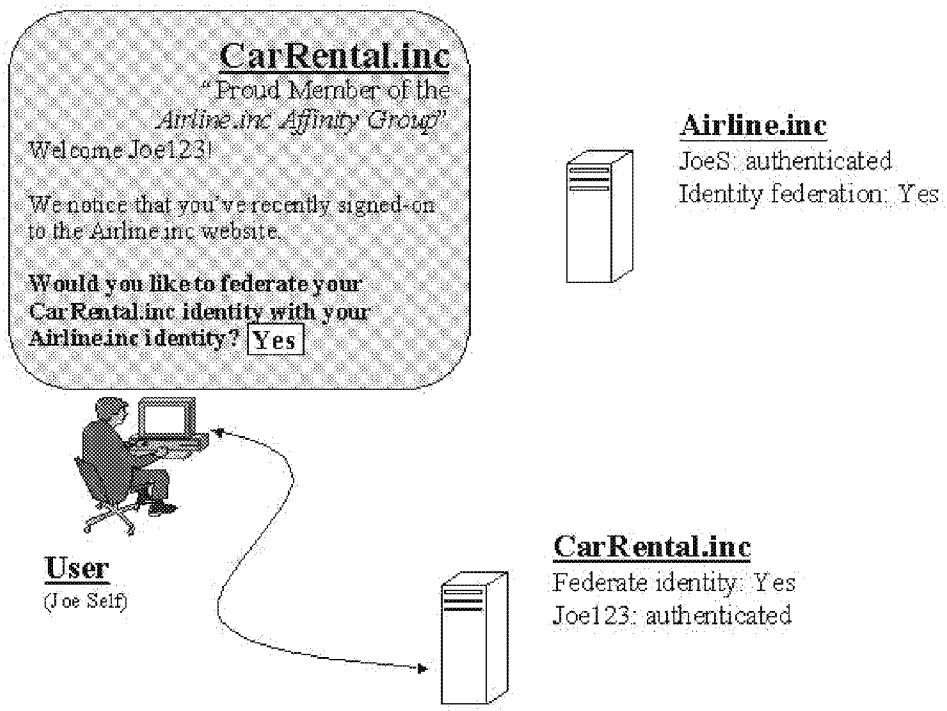
176 If the service provider maintains local accounts, as in our example, it will typically, upon Joe Self's arrival, prompt
177 Joe to log in, which he does using his local CarRental.inc identity. See Figure 5.



178

179 **Figure 5. User signs-on using his local service provider identity.**

180 Thereafter, Joe Self is presented with the opportunity to federate his local identities between CarRental.inc and
181 Airline.inc. See Figure 6.



182

183 **Figure 6. User is prompted to federate his local identities and selects "yes."**

184 **POLICY/SECURITY NOTE:**

185 Whether the service provider asks for consent to federate the user's local identity before or after locally
186 authenticating the user is a matter of local deployment policy.

187 As a part of logging in to the CarRental.inc Website, Joe Self's local CarRental.inc identity is federated with his local
188 Airline.inc identity. See Figure 7.

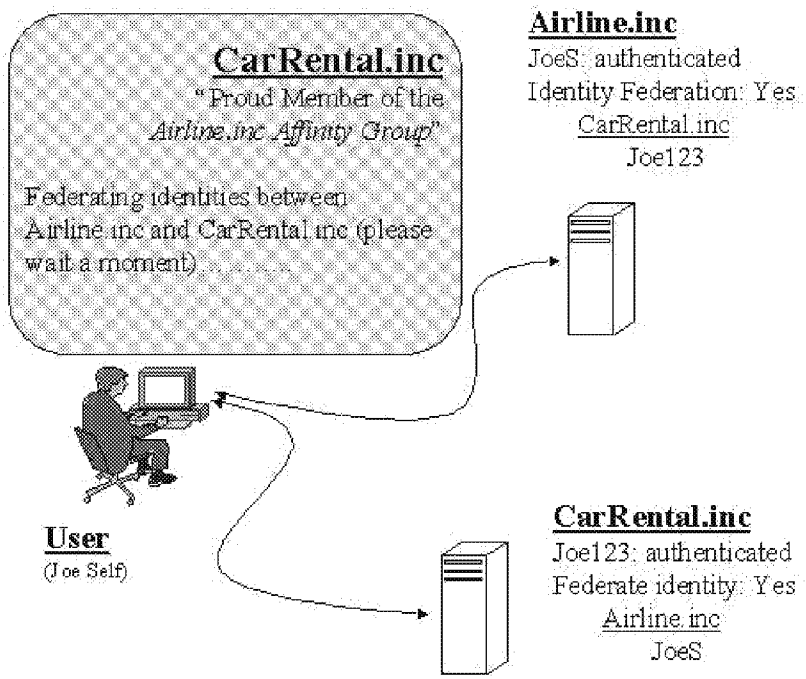


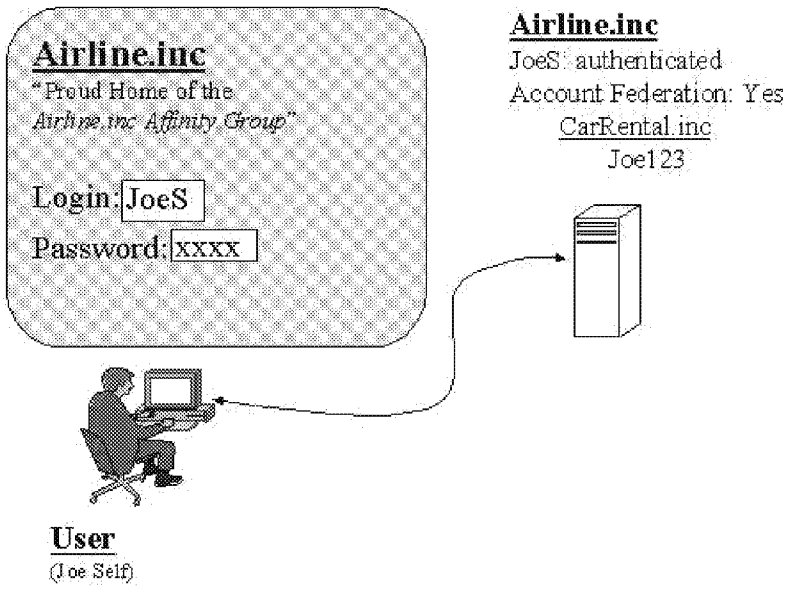
Figure 7. The Websites federate the user's local identities.

Upon completion of the login and identity federation activity, Joe User is logged in to the CarRental.inc Website, and CarRental.inc delivers services to him as usual. In addition, the Website may now offer new selections because Joe Self's local service provider (CarRental.inc) identity has been federated with his local identity provider (Airline.inc) identity. See Figure 8.

TECHNICAL NOTE:

Some figures illustrating the user experience, for example, Figure 7, show simplified, user-perspective notions of how identity federation is effected. In actuality, cleartext identifiers, for example, "JoeS" and "Joe123" WILL NOT be exchanged between the identity provider and service provider. Rather, opaque user handles will be exchanged. See Section 4.4.1 for details.

Additionally, if errors are encountered in the process of authenticating and/or federating, the service provider will need to present appropriate indications to the user.



202

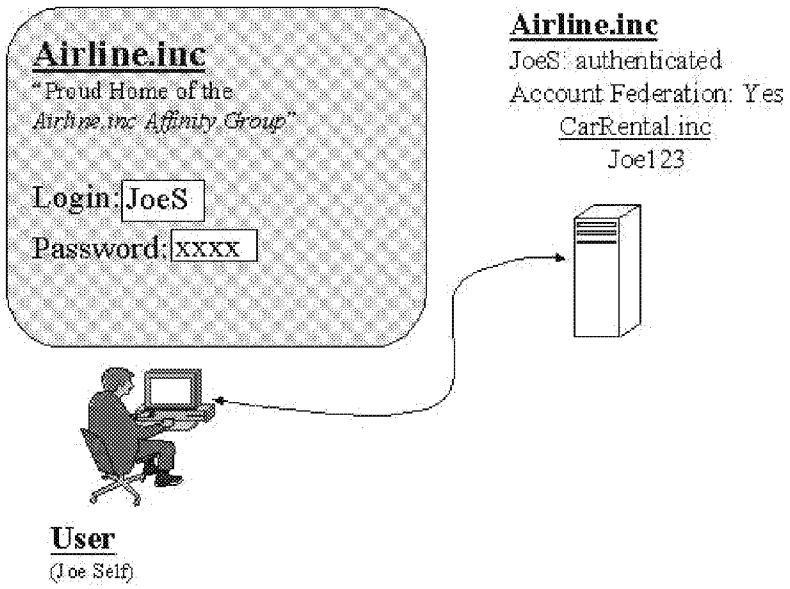
203 **Figure 8. The service provider delivers services to user as usual.**

204 **POLICY/SECURITY NOTE:**

205 Business prerequisites must be met to offer identity federation. Two prerequisites are notifying the user of
206 the capability to federate and soliciting consent to facilitate introductions. Another is creating agreements
207 between the affinity group members to establish their policies for recognizing identities and honoring
208 reciprocal authentication.

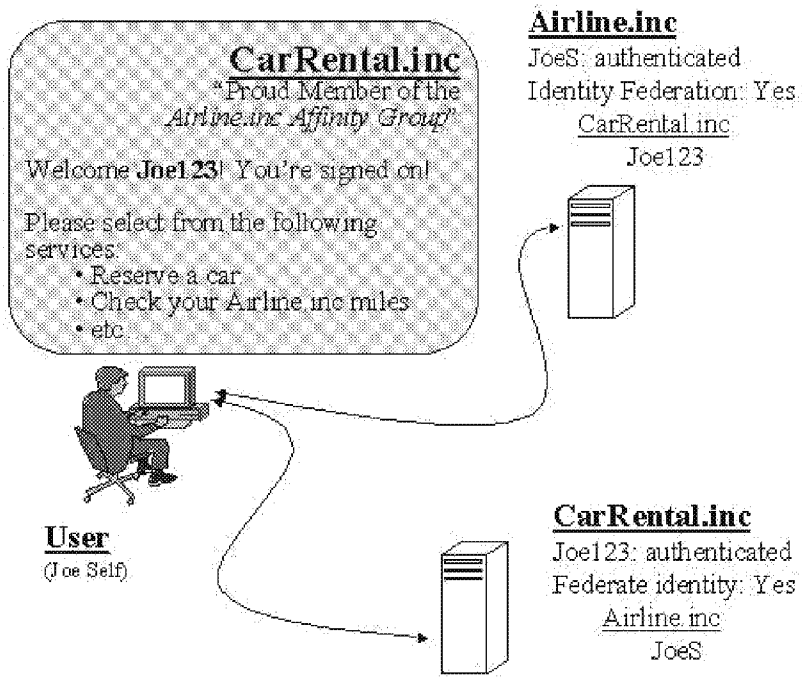
209 2.2. Example of Single Sign-on User Experience

210 Single sign-on builds upon identity federation and has a simple user experience. Joe Self logs in to the Airline.inc
211 Website and later visits the CarRental.inc Website with which he has established identity federation. Joe Self's
212 authentication state with the Airline.inc Website is reciprocally honored by the CarRental.inc Website, and Joe Self is
213 transparently logged in to the latter site. See Figure 9 and Figure 10.



214

215 **Figure 9. User logs in to identity provider's Website using local identity.**



216

217 **Figure 10. User proceeds to service provider's Website, and his authentication state is reciprocally honored by the service**
218 **provider's Website.**

219 A perceptive Joe Self will notice that his name in the CarRental.inc session is based upon his local CarRental.inc
220 identity, rather than the local Airline.inc identity with which it has been federated.

TECHNICAL NOTE:

Because users' actual account identifiers are not exchanged during federation, a service provider will not be able to display a user's identity provider identifier.

Also, many types of service provider Websites may not use a personally identifiable identifier in response to the user. For example, advertising-driven sites where users may specify display preferences, for example, a sporting events schedule site. The site may simply transparently refer to the user as "you," for example, "Set your display preferences here...," "Here is the list of upcoming events you're interested in...," etc.

SECURITY/POLICY NOTE:

Even though the user may be validly authenticated via the single sign-on mechanism, the user's use of the service provider's Website is still subject to local policy. For example, the site may have time-of-day usage restrictions, the site may be undergoing maintenance, the user's relationship with the service provider may be in a particular state (for example, highly valued customer - show the user the bonus pages; troublesome customer - remind the user of unpaid bills and restrict some access).

3. Liberty Engineering Requirements Summary

This section summarizes the Liberty general and functional engineering requirements.

3.1. General Requirements

The Liberty-enabled systems should follow the set of general principals outlined in Section 3.1.1 and Section 3.1.2. These principles cut across categories of functionality.

3.1.1. Client Device/User Agent Interoperability

Liberty Version 1.2 clients encompass a broad range of presently deployed Web browsers, other presently deployed Web-enabled client access devices, and newly designed Web-enabled browsers or clients with specific Liberty-enabled features.

The Liberty Version 1.2 architecture and protocol specifications must support a basic level of functionality across the range of Liberty Version 1.2 clients.

3.1.2. Openness Requirements

The Liberty architecture and protocol specifications must provide the widest possible support for:

- Operating systems
- Programming languages
- Network infrastructures

and must not impede multivendor interoperability between Liberty clients and services, including interoperability across circle of trust boundaries.

3.2. Functional Requirements

The Liberty architecture and protocols must be specified so that Liberty-enabled implementations are capable of performing the following activities:

- Identity federation
- Authentication
- Use of pseudonyms
- Support for Anonymity
- Global logout

3.2.1. Identity Federation

Requirements of identity federation stipulate that:

- Providers give the user notice upon identity federation and defederation.

- Service providers and identity providers notify each other about identity defederation.
- Each identity provider notifies appropriate service providers of user account terminations at the identity provider.
- Each service provider and/or identity provider gives each of its users a list of the user's federated identities at the identity provider or service provider.
- A service provider may also request an anonymous, temporary identity for a Principal.

3.2.2. Authentication

Authentication requirements include:

- Supporting any method of navigation between identity providers and service providers on the part of the user, that is, how the user navigates from A to B (including click-through, favorites or bookmarks, URL address bar, etc.) must be supported.
- Giving the identity provider's authenticated identity to the user before the user gives credentials or any other personally identifiable information to the identity provider.
- Providing for the confidentiality, integrity, and authenticity of information exchanged between identity providers, service providers, and user agents, as well as mutually authenticating the identities of the identity providers and service providers, during the authentication and single sign-on processes.
- Supporting a range of authentication methods, extensibly identifying authentication methods, providing for coalescing authentication methods into authentication classes, and citing and exchanging authentication classes. Protocols for exchanging this information are out of the scope of the Liberty Version 1.2 specifications, however.
- Exchanging the following minimum set of authentication information with regard to a user: authentication status, instant, method, and pseudonym (which may be temporary or persistent).
- Giving service providers the capability of causing the identity provider to reauthenticate the user using the same or a different authentication class. Programmatic exchange of the set of authentication classes for which a user is registered at an identity provider is out of the scope of the Liberty Version 1.2 specifications, however.
- Allowing an identity provider, at the discretion of the service provider, to authenticate the user via an identity provider other than itself and relay this information to a service provider.

3.2.3. Pseudonyms

Liberty-enabled implementations must be able to support the use of pseudonyms that are unique on a per-identity-federation basis across all identity providers and service providers.

3.2.4. Anonymity

A service provider may request that an identity provider supply a temporary pseudonym that will preserve the anonymity of a Principal. This identifier may be used to obtain information for or about the Principal (with his or her permission) via mechanisms that are outside the scope of the ID-FF, without requiring the user to consent to a long term relationship with the service provider.

3.2.5. Global Logout

Liberty-enabled implementations must be able to support the notification of service providers when a user logs out at identity provider.

4. Liberty Architecture

The overall Liberty architecture is composed of three orthogonal architectural components (see Figure 11):

- Web redirection
- Web services
- Metadata and schemas

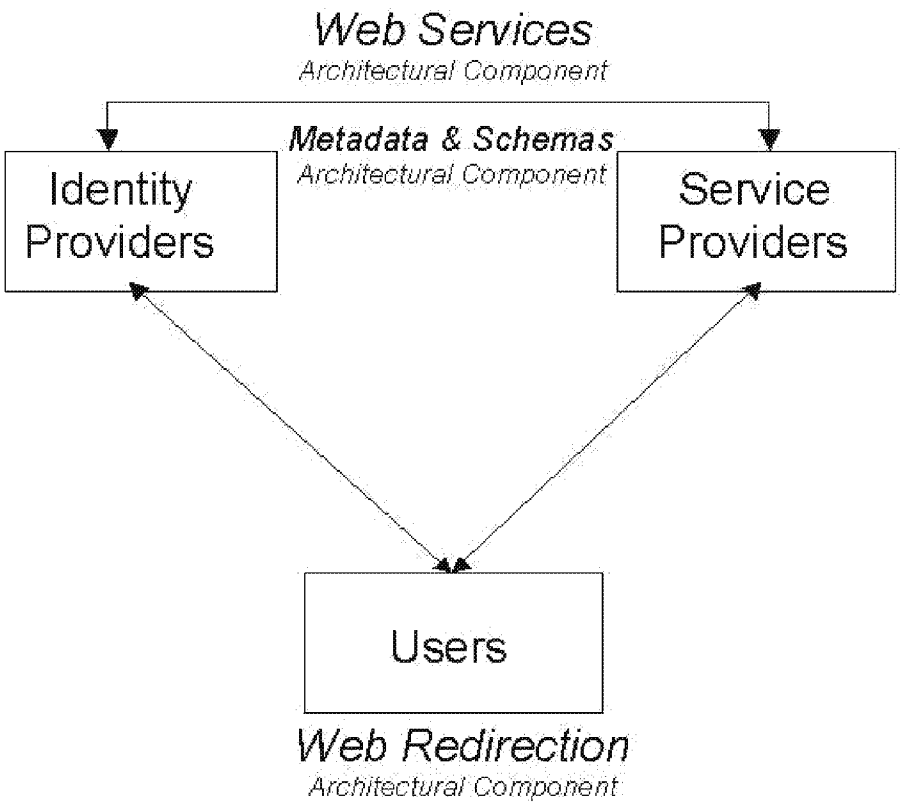


Figure 11. Overall Liberty architecture

The role of each architectural component is summarized in Table 2:

Table 2. Components of Liberty architecture

Web redirection	Action that enables Liberty-enabled entities to provide services via today's user-agent-installed base.
Web services	Protocol profiles that enable Liberty-enabled entities to directly communicate.
Metadata and schemas	A common set of metadata and formats used by Liberty-enabled sites to communicate various provider-specific and other information.

Section 4.1 through Section 4.3 describe each architectural component. Section 4.4 through Section 4.6 then relate the architectural components to the concrete protocols and profiles detailed in [LibertyProtSchema] and [LibertyBindProf], and Section 4.7 provides illustrations of user experience.

4.1. Web Redirection Architectural Component

The Web redirection architectural component is composed of two generic variants: HTTP-redirect-based redirection and form-POST-based redirection. Both variants create a communication channel between identity providers and service providers that is rooted in the user agent. See Figure 12.

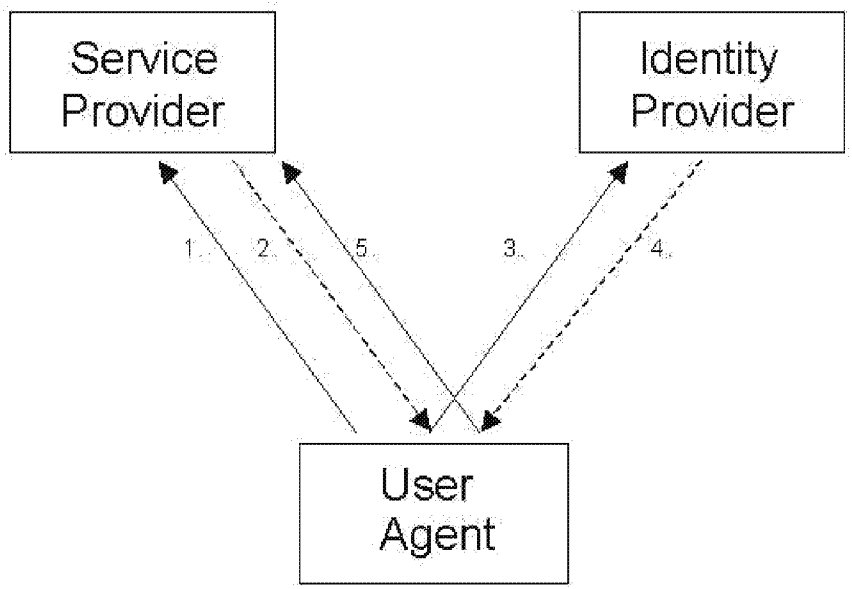


Figure 12. Web redirection between a service provider and an identity provider via the user agent

4.1.1. HTTP-Redirect-Based Redirection

HTTP-redirect-based redirection uses the HTTP redirection class of response (that is, *redirects*) of the HTTP protocol (see [RFC2616]) and the syntax of URIs (see [RFC1738] and [RFC2396]) to provide a communication channel between identity providers and service providers. Thus the steps shown in Figure 12 create a communication channel between the service provider and identity provider as follows:

1. The user agent sends an HTTP request to the service provider (typically a GET). In this step the user has typically clicked on a link in the Webpage presently displayed in the user agent.

2. The service provider responds with an HTTP response with a status code of 302 (that is, a redirect) and an alternate URI in the Location header field. In this example, the Location URI will point to the identity provider and will also contain a second, embedded URI pointing back to the service provider.

3. The user agent sends an HTTP request to the identity provider (typically a GET), specifying the complete URI taken from the Location field of the response returned in Step 2 as the argument of the GET. Note: This URI contains the second, embedded URI pointing back to the service provider.

4. The identity provider can then respond in kind with a redirect whose Location header field contains the URI pointing to the service provider (extracted from the GET argument URI supplied in Step 3) and optionally contains an embedded, second URI pointing back to itself.

5. The user agent sends an HTTP request to the service provider (typically a GET), specifying the complete URI taken from the Location field of the response returned in Step 4 as the argument of the GET. Note: This URI might contain any second, embedded URI pointing back to the identity provider.

Note:

Both URIs are passed as arguments of HTTP GET requests, and the Location response-header field of redirect responses can contain either or both embedded URIs and other arbitrary data. Thus the identity provider and service provider can relatively freely exchange arbitrary information between themselves across this channel. See Table 3.

Table 3. Embedding a parameter within an HTTP redirect

Location: http://www.foobar.com/auth	Redirects to foobar.com
Location: http://www.foobar.com/auth?XYZ=1234	Redirects to foobar.com and also passes a parameter "XYZ" with the value "1234"

4.1.2. Form-POST-Based Redirection

In form-POST-based redirection, the following steps in Figure 12 are modified as follows:

2. The service provider responds by returning an HTML form to the user agent containing an action parameter pointing to the identity provider and a method parameter with the value of POST. Arbitrary data may be included in other form fields. The form may also include a JavaScript or ECMAscript fragment that causes the next step to be performed without user interaction.

3. Either the user clicks on the Submit button, or the JavaScript or ECMAscript executes. In either case, the form and its arbitrary data contents are sent to the identity provider via the HTTP POST method.

The above process can be reversed in Steps 4 and 5 to effect form-POST-based communication in the opposite direction.

4.1.3. Cookies

POLICY/SECURITY NOTE:

Use of cookies by implementors and deployers should be carefully considered, especially if a cookie contains either or both personally identifying information and authentication information. Cookies can be either ephemeral (that is, this session only) or persistent. Persistent cookies are of special concern because they are typically written to disk and persist across user agent invocations. Thus if a session authentication token is cached in a persistent cookie, the user exits the browser, and another person uses the system and relaunches the browser, then the second person could impersonate the user (unless any authentication time limits imposed by the authentication mechanism have expired).

Additionally, persistent cookies should be used *only* with the consent of the user. This consent step allows, for example, a user at a public machine to prohibit a persistent cookie that would otherwise remain in the user agent's cookie cache after the user is finished.

4.1.3.1. Why Not Use Cookies in General?

Cookies are the HTTP state management mechanism specified in [RFC2965] and are a means for Web servers to store information, that is, maintain state, in the user agent. However, the default security setting in the predominant user agents allow cookies to be read only by the Website that wrote them. This discrimination is based on the DNS domains of the reading and writing sites.

To permit multiple identity providers and service providers in different DNS domains to communicate using cookies, users must lower the default security settings of their user agents. This option is often an unacceptable requirement.

Additionally, it is not uncommon for users and/or their organizations to operate their user agents with cookies turned off.

4.1.3.2. Where Cookies are Used

In the Liberty context, cookies might be used for maintaining local session state, and cookies are used in addressing the introduction problem (see Section 4.5).

The fact that identity providers cannot arbitrarily send data to service providers via cookies does not preclude identity providers and service providers from writing cookies to store local session state and other, perhaps persistent, information.

4.1.4. Web Redirection Summary

Web redirection is not an ideal distributed systems architecture.

POLICY/SECURITY NOTE:

Communications across Web redirection channels as described in Section 4.1.1 through Section 4.1.3 have many well-documented security vulnerabilities, which should be given careful consideration when designing protocols utilizing Web redirection. Such consideration was incorporated into the design of the profiles specified in [LibertyBindProf], and specific considerations are called out as appropriate in that document (for example, regarding cleartext transmissions and caching vulnerabilities). Examples of security vulnerabilities include:

- **Interception:** Such communications go across the wire in cleartext unless all the steps in Section 4.1.1 through Section 4.1.3 are carried out over an SSL or TLS session or across another secured communication transport, for example, an IPsec-based VPN.

- **User agent leakage:** Because the channel is redirected through the user agent, many opportunities arise for the information to be cached in the user agent and revealed later. This caching is possible even if a secure transport is used because the conveyed information is kept in the clear in the browser. Thus any sensitive information conveyed in this fashion needs to be encrypted on its own before being sent across the channel.

TECHNICAL NOTE:

A key limitation of Web redirection is the overall size of URIs passed as arguments of GET requests and as values of the Location field in redirects. These elements have size limitations that vary from browser to browser and are particularly small in some mobile handsets. These limitations were incorporated into the design of the protocols specified in [LibertyProtSchema] and [LibertyBindProf].

In spite of the vulnerabilities and limitations of Web redirection, use of this mechanism enables distributed, cross-domain interactions, such as single sign-on, with today's deployed HTTP infrastructure on the Internet.

Both generic variants of Web redirection underlie several of the profiles specified in [LibertyBindProf]: Single Sign-On and Federation, Identity Federation Termination Notification, Name Identifier Registration, and Single Logout.

4.2. Web Services Architectural Component

Various Liberty protocol interaction steps are profiled to occur directly between system entities in addition to other steps occurring via Web redirection and are based on RPC-like protocol messages conveyed via SOAP (see [SOAPv1.1]). SOAP is a widely implemented specification for RPC-like interactions and message communications using XML and HTTP and hence is a natural fit for this architectural component.

4.3. Metadata and Schemas Architectural Component

Metadata and schemas is an umbrella term generically referring to various subclasses of information and their formats exchanged between service providers and identity providers, whether via protocol or out of band. The subclasses of exchanged information are

- **Account/Identity:** In Liberty Version 1.2, account/identity is simply the opaque user handle that serves as the name that the service provider and the identity provider use in referring to the user when communicating. In other Liberty phases, it encompasses various attributes.

- **Authentication Context:** Liberty explicitly accommodates identity provider use of arbitrary authentication mechanisms and technologies. Different identity providers will choose different technologies, follow different processes, and be bound by different legal obligations with respect to how they authenticate users. The choices that an identity provider makes here will be driven in large part by the requirements of the service providers with which the identity provider has federated. Those requirements, in turn, will be determined by the nature of the service (that is, the sensitivity of any information exchanged, the associated financial value, the service providers risk tolerance, etc) that the service provider will be providing to the user. Consequently, for anything other than trivial services, if the service provider is to place sufficient confidence in the authentication assertions it receives from an identity provider, the service provider must know which technologies, protocols, and processes were used or followed for the original authentication mechanism on which the authentication assertion is based. The authentication context schema provides a means for service providers and identity providers to communicate such information (see [LibertyAuthnContext]).

- **Provider Metadata:** For identity providers and service providers to communicate with each other, they must a priori have obtained metadata regarding each other. These provider metadata include items such as X.509 certificates and service endpoints. [LibertyMetadata] defines metadata schemas for identity providers and service providers that may be used for provider metadata exchange.

4.4. Single Sign-On and Identity Federation

The single sign-on and identity federation aspects of Liberty are facilitated by the Single Sign-On and Federation Protocol, which is specified in [LibertyProtSchema]. It facilitates both identity federation (see Section 4.4.1) and single sign-on (see Section 4.4.2) in a single overall protocol flow. The various profiles of the overall protocol flow that are defined in [LibertyBindProf] are discussed in Section 4.4.3.

4.4.1. Single Sign-On and Identity Federation

The first time that users use an identity provider to log in to a service provider they must be given the option of federating an existing local identity on the service provider with the identity provider login to preserve existing information under the single sign-on. See Figure 13. It is critical that, in a system with multiple identity providers and service providers, a mechanism exists by which users can be (at their discretion) uniquely identified across the providers. However, it is technically challenging to create a globally unique ID that is not tied to a particular identity provider and a business challenge to ensure the portability of globally unique IDs.

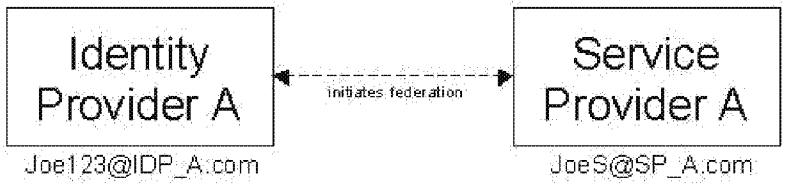


Figure 13. User initiates federation of two identities

An explicit trust relationship, or chain, is created with the opt-in identity federation that occurs the first time a user logs in to a service provider using an identity provider. While multiple identities can be federated to each other, an explicit link exists between each identity. Providers cannot skip over each other in the trust chain to request information on or services for a user because user identity information must be checked at each step. Therefore, the only requirement is that, when two elements of a trust chain communicate, they can differentiate users.

Members of the circle of trust are not required to provide the actual account identifier for a user and can instead provide a handle for a particular user. Members can also choose to create multiple handles for a particular user. However, identity providers must create a single handle for each service provider that has multiple Websites so that the handle can be resolved across the Websites.

Because both the identity provider and service provider in such a federation need to remember the other's handle for the user, they create entries in their user directories for each other and note each other's handle for the user. See Figure 14 and Figure 15.

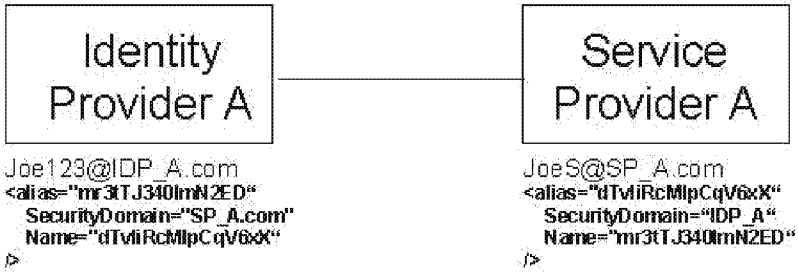


Figure 14. User directories of the identity provider and service provider upon identity federation

TECHNICAL NOTE:

Figure 14, along with the three following figures, illustrate bilateral identity federation; this is where both the service provider and identity provider exchange handles for the user. However, bilateral handle exchange

is an *optional* feature of the Liberty Single Sign-On and Federation protocol. In some scenarios, only the identity provider's handle will be conveyed to the service provider(s). This will typically be the case where the service provider doesn't otherwise maintain its own user repository.

The lines connecting the identity and service providers in the aforementioned figures signify federation relationships rather than communication exchanges.

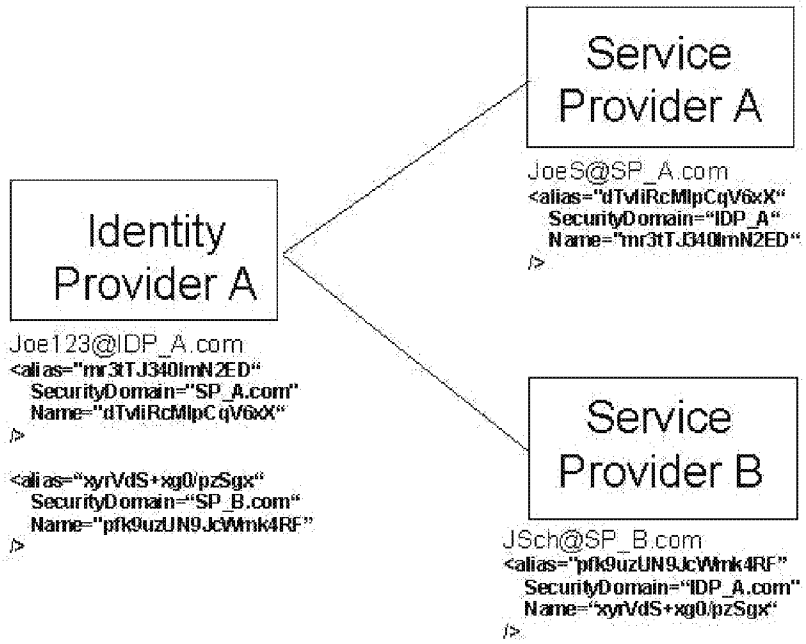


Figure 15. User directories of the identity provider and multiple service providers upon identity federation.

POLICY/SECURITY NOTE:

1. Observe in Figure 15 that SP_A and SP_B cannot communicate directly about Joe Self. They can only communicate with the identity provider individually. This feature is desirable from policy and security perspectives. If Joe Self wishes the service providers to be able to exchange information about him, then he must explicitly federate the two service provider identities, effectively opting in.
Another aspect of this feature is that if the user's local identity is compromised on, for example, SP_A, the local identities at IDP_A or SP_B are not necessarily also compromised.

2. Properties of the user handles, for example, mr3tTJ340ImN2ED, (also known as *name identifiers*) need to be carefully considered. It may not be enough for them to be opaque. Considerations of the construction of name identifiers are discussed in [LibertyProtSchema]. Additionally, user handles should be refreshed periodically. Service providers may refresh the user handles they optionally supply to identity providers via the register name identifier profile defined in [LibertyBindProf]. Identity providers may also use the same profile to optionally refresh the user handles they supply to service provides.

While it is obvious that a user can sign in at multiple service providers with an identity provider, a user can also link multiple identity providers to a particular service provider. See Figure 16. This ability proves useful when a user switches from a work computer to a home computer or from a computer to a mobile device, each of which may be associated with a different identity provider and circle of trust.

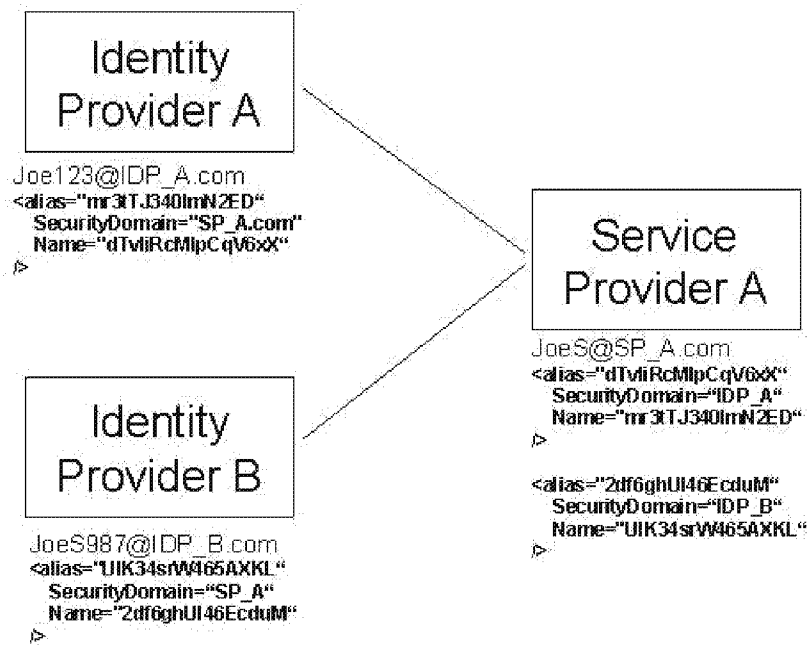


Figure 16. A user with two identity providers federated to a service provider

POLICY/SECURITY NOTE:

Subtle considerations arise here in terms of how easy it is for a user to switch between identities and how this capability is materialized. IDP_A may belong to the same circles of trust as more than one of the user's devices. Therefore, certain questions arise, for example, How do users know to which (or both) identity provider they are presently logged in? Features satisfying such questions are a way for identity providers and circles of trust to differentiate themselves.

While federating two identity providers to a service provider, as illustrated in Figure 16, enables the user to log in to the service provider using either identity provider, the user must remember to federate new service providers to both identity providers, which can be a cumbersome process. An alternative is for the user to federate identity providers together and set policies enabling identity providers to access each other's information. See Figure 17 and the following POLICY/SECURITY NOTE. The user can then use a preferred identity provider to log in to service providers, but always has the choice of adding additional identity providers to a service provider.

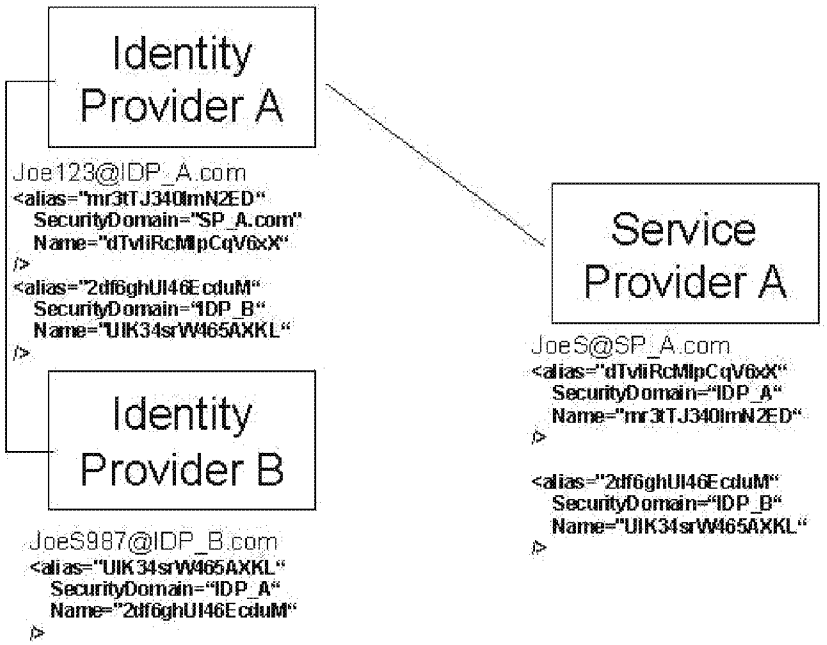


Figure 17. A user with two identity providers federated

TECHNICAL NOTE:

In Figure 17, Identity Provider A is acting as both a service provider and an identity provider.

POLICY/SECURITY NOTE:

• The semantics of such a federated relationship (Figure 17) between identity providers are not dictated by the underlying Liberty protocols, nor are they precluded. These semantics need to be addressed by the agreements between the identity providers and supported by the capabilities of the deployed Liberty-enabled implementations.

• Additionally, how trust relationships between identity providers are established, and how those relationships are represented to service providers, are unspecified. Identity providers enabling relationships such as that illustrated in Figure 17 must mutually define governing policies and means of representing such trust relationships to relying service providers (for example Service Provider A in Figure 17).

• Circle of trust agreements should address how federation failures are materialized to users.

• Appropriate portions of the assertions passed between the identity provider and the service provider to effect federation should be logged.

• By creating many local identities with many service providers and/or identity providers and then federating them, users possess many sets of local credentials that may be used as a basis to authenticate with many service providers via single sign-on. This situation constitutes a risk. For example, every identity provider that possesses reusable user credentials, for example, a username and password, can impersonate the user at every service provider federated with that account.
In the normal course of events, some local credentials may go unused for periods of time because the user is making use of the local account via single sign-on from another identity provider. Thus a means of controlling the growth of a user's set of local credentials might be to offer the user the option of invalidating local credentials at identity federation time and also perhaps after a certain number of times of visiting the Website without using them.

4.4.1.1. No Need for Global Account/Identity Namespace

Given the above architecture where users opt to federate identities at different identity providers and service providers, a global namespace across all of the players should not be needed. Circle of trust members can communicate with each other, about or on a user's behalf, only when a user has created a specific federation between the local identities and has set policies for that federation. Although long chains of identity providers and service providers can be created, the user's identity is federated in each link in the chain and, therefore, a globally unique ID need not exist for that user across all of the elements of the chain. See Figure 17.

4.4.1.2. Single Sign-On with Anonymity

In some scenarios, a user may not need to establish a long term relationship or identifier with a service in order to use that service, or gain the benefits of single sign-on across services using the same identity provider. Typically, the short-term identifier that is given to a service can be leveraged at the time of sign-on to obtain other information or provide services to the user through the use of additional protocols that are outside the scope of Liberty ID-FF.

POLICY/SECURITY NOTE:

When such an identifier is requested, it must be generated for a single use, and given only to a single service provider, rather than shared or reused. Other information shared about the user through other means should be at the user's discretion.

4.4.1.3. Federation Management: Defederation

Users will have the ability to terminate federations, or defederate identities. [LibertyProtSchema] and [LibertyBind-Prof] specify a Federation Termination Notification Protocol and related profiles. Using this protocol, a service provider may initiate defederation with an identity provider or vice versa. The nominal user experience is for the user to select a Defederate link on a service provider's or identity provider's Webpage. This link initiates defederation with respect to some other, specific, identity provider or service provider.

When defederation is initiated at an identity provider, the identity provider is stating to the service provider that it will no longer provide user identity information to the service provider and that the identity provider will no longer respond to any requests by the service provider on behalf of the user.

When defederation is initiated at a service provider, the service provider is stating to the identity provider that the user has requested that the identity provider no longer provide the user identity information to the service provider and that service provider will no longer ask the identity provider to do anything on the behalf of the user.

POLICY/SECURITY NOTE:

Regarding defederation, several issues must be considered:

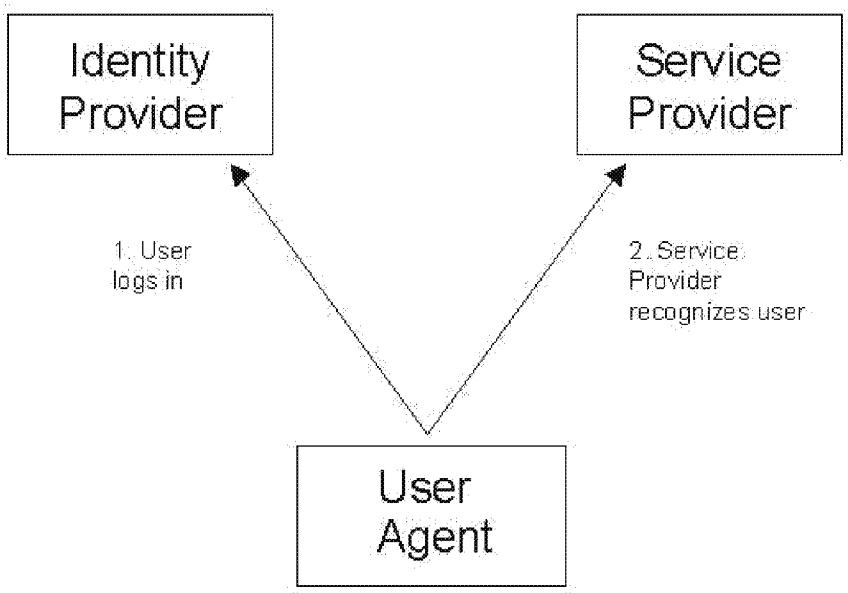
- The user should be authenticated by the provider at which identity defederation is being initiated.
- Providers should ask the user for confirmation before performing defederation and appropriately log the event and appropriate portions of the user's authentication information.
- It is recommended that the service provider, after initiating or receiving a federation termination notification for a Principal, check whether that Principal is presently logged in to the service provider on the basis of an assertion from the identity provider with which the federation termination notification was exchanged. If so, then the local session information that was based on the identity provider's assertion should be invalidated. If the service provider has local session state information for the Principal that is not based on assertions made by the identity provider with which the federation termination notification was exchanged, then the service provider may continue to maintain that information.

567 • If the Principal subsequently initiates a single sign-on session with the same identity provider, the service
568 provider will need to request federation as well as authentication from the identity provider.

569 • Other means of federation termination are possible, such as federation expiration and termination of
570 business agreements between service providers and identity providers.

571 4.4.2. Single Sign-on

572 Single sign-on is enabled once a user's identity provider and service provider identities are federated. From a user's
573 perspective, single sign-on is realized when the user logs in to an identity provider and uses multiple affiliated service
574 providers without having to sign on again (see Figure 18). This convenience is accomplished by having federated
575 the user's local identities between the applicable identity providers and the service providers. The basic user single
576 sign-on experience is illustrated in Section 4.4.1.



577

578 **Figure 18. User logs in at identity provider and is recognized by service provider**

579 [LibertyBindProf] specifies single sign-on by profiling both the "Browser/Artifact Profile" and the "Browser/Post
580 Profile" of SAML (see [SAMLBind]).

581 **Note:**

582 POLICY/SECURITY NOTE: Regarding authentication, single sign-on, credentials, etc., several issues must
583 be considered:

584 **Authentication Mechanisms are Orthogonal to Single Sign-On** Single sign-on is a means by which
585 a service provider or identity provider may convey to another service provider or
586 identity provider that the user is in fact authenticated. The means by which the user
587 was originally authenticated is called the authentication mechanism. Examples of
588 authentication mechanisms are username with password (*not* HTTP Basic Auth),
589 certificate-based (for example, via SSL or TLS), Kerberos, etc.

Identity Provider Session State Maintenance Identity providers need to maintain authentication state information for principals. This is also known as "local session state maintenance", where "local" implies "local to the identity provider". There are several mechanisms for maintaining local session state information in the context of HTTP-based [RFC2616] user agents (commonly known as "web browsers"). Cookies are one such mechanism and are specified in [RFC2965]. Identity providers use local session state information, mapped to the participating user agent (see Figure 18), as the basis for issuing authentication assertions to service providers who are performing the "Single Sign-On and Federation" protocol [LibertyBindProf] with the identity provider. Thus, when the Principal uses his user agent to interact with yet another service provider, that service provider will send an `<AuthnRequest>` to the identity provider. The identity provider will check its local session state information for that user agent, and return to the service provider an `<AuthnResponse>` containing an authentication assertion if its local session state information indicates the user agent's session with the identity provider is presently active.

Credentials Credentials are relied upon in a number of ways in a single sign-on system and are often the basis for establishing trust with the credential bearer. Credentials may represent security-related attributes of the bearer, including the owner's identity. Sensitive credentials that require special protection, such as private cryptographic keys, must be protected from unauthorized exposure. Some credentials are intended to be shared, such as public-key certificates.

Credentials Credentials are a general notion of the data necessary to prove an assertion. For example, in a password-based authentication system, the user name and password would be considered credentials. However, the use of credentials is not limited to authentication. Credentials may also be relied upon in the course of making an authorization decision.

As mentioned above, certain credentials must be kept confidential. However, some credentials not only need to remain confidential, but also must be integrity-protected to prevent them from being tampered with or even fabricated. Other credentials, such as the artifacts described in Section 4.4.3.1, must have the properties of a nonce. A nonce is a random or nonrepeating value that is included in data exchanged by a protocol, usually for guaranteeing liveness and thus detecting and protecting against replay attacks.

Authentication Type, Multitiered Authentication All authentication assertions should include an authentication type that indicates the quality of the credentials and the mechanism used to vet them. Credentials used to authenticate a user or supplied to authorize a transaction and/or the authentication mechanism used to vet the credentials may not be of sufficient quality to complete the transaction.

For example, a user initially authenticates to the identity provider using username and password. The user then attempts to conduct a transaction, for instance, a bank withdrawal, which requires a stronger form of authentication. In this case the user must present a stronger assertion of identity, such as a public-key certificate or something ancillary such as birthdate, mother's maiden name, etc. This act is *reauthentication* and the overall functionality is *multitiered authentication*. Wielding multitiered authentication can be a policy decision at the service provider and can be at the discretion of the service provider. Or it might be established as part of the contractual arrangements of the circle of trust. In this case, the circle of trust members can agree among themselves upon the trust they put in different authentication types and of each other's authentication assertions. Such an agreement's form may be similar to today's certificate practice statements (CPS) (for example, see http://www.verisign.com/repository/cps20/cps20.pdf). The information cited in such a document may include

- User identification methods during credentials enrollment
- Credentials renewal frequency
- Methods for storing and protecting credentials (e.g., smartcard, phone, encrypted file on hard drive)

Note:
While the current Liberty specifications allow service providers, identity providers, and user agents to support authentication using a range of methods, the methods and their associated protocol exchanges are not specified within Liberty documents. Further, the scope of the current Liberty specifications does not include a means for a communicating identity provider and user agent to identify a set of methods that they are both equipped to support. As a result, support for the Liberty specifications is not in itself sufficient to ensure effective interoperability between arbitrary identity providers and user agents using arbitrary methods and must, instead, be complemented with data obtained from other sources.

Also, the scope of the current Liberty specifications does not include a means for a service provider to interrogate an identity provider and determine the set of authentication profiles for which a user is registered at that identity provider. As a result, effective service provider selection of specific profiles to authenticate a particular user will require access to out-of-band information describing users' capabilities.

For example, members of a given circle of trust may agree that they will label an authentication assertion based on PKI technology and face-to-face user identity verification with substantiating documentation at enrollment time to be of type "Strong." Then, when an identity provider implementing these policies and procedures asserts that a user has logged in using the specified PKI-based authentication mechanism, service providers rely upon said assertion to a certain degree. This degree of reliance is likely different from the degree put into an assertion by an identity provider who uses the same PKI-based authentication mechanism, but who does not claim to subject the user to the same amount of scrutiny at enrollment time.This issue has another dimension: Who performs the reauthentication? An identity provider or the service provider itself? This question is both an implementation and deployment issue and an operational policy issue. Implementations and deployments need to support having either the identity provider or the service provider perform reauthentication when the business considerations dictate it (that is, the operational policy). For example, a circle of trust may decide that the risk factors are too large for having the identity provider perform reauthentication in certain high-value interactions and that the service provider taking on the risk of the interaction must be able to perform the reauthentication.

Mutual Authentication Another dimension of the authentication type and quality space is mutual authentication. For a user authenticating himself to an identity provider, mutual authentication implies that the identity provider server authenticates itself with the user as well as vice versa. Mutual authentication is a function of the particular authentication mechanism employed. For example, any user authentication performed over SSL or TLS is mutual authentication because the server is authenticated to the client by default with SSL or TLS. This feature can be the basis of some greater assurance, but does have its set of vulnerabilities. The server may be wielding a bogus certificate, and the user may not adequately inspect it or understand the significance.

Validating Liveness *Liveness* refers to whether the user who authenticated at time *t0* is the same user who is about to perform a given operation at time *t1*. For example, a user may log in and perform various operations and then attempt to perform a given operation that the service provider considers high-value. The service provider may initiate reauthentication to attempt to validate that the user operating the system is still the same user that authenticated originally. Even though such an approach has many vulnerabilities, that is, it fails completely in the case of a rogue user, it does at least augment the service provider's audit trail. Therefore, at least some service providers will want to do it.

Authentication assertions from identity providers contain a `<ReauthenticationOnOrAfter>` element. If this attribute was specified and the time of the user request is past the specified reauthentication time, the service provider should redirect the user back to the identity provider for reauthentication.

Communication Security A service provider can reject communications with an identity provider for various reasons. For example, it may be the policy of a service provider to require that all protocol exchanges between it and the bearer of a credential commence over a communication protocol that has certain qualities such as bilateral authentication, integrity protection, and message confidentiality.

4.4.3. Profiles of the Single Sign-On and Federation Protocol

The Single Sign-On and Federation Protocol, as specified in [LibertyProtSchema], defines messages exchanged between service providers and identity providers. The concrete mapping of these messages to particular transfer (for example, HTTP) and/or messaging (for example, SOAP) protocols and precise protocol flows are specified in [LibertyBindProf]. These mappings are called profiles. The Single Sign-On and Federation Protocol specifies three profiles. The following sections summarize each profile. For a detailed discussion of the common interactions and processing rules of these profiles and for details about each profile, see [LibertyBindProf].

TECHNICAL NOTE:

The Single Sign-On and Federation Protocol and related profiles specify means by which service providers indicate to identity providers the particular profile they wish to employ. The primary means is the `<lib:ProtocolProfile>` element of the `<lib:AuthnRequest>` message, which is employed by all profiles of the Single Sign-On and Federation Protocol. Note: The Liberty-enabled client and proxy profile employs additional means.

4.4.3.1. Liberty Artifact Profile

The Liberty artifact profile specifies embedding an artifact in a URI exchanged between the identity provider and service provider via Web redirection and also requires direct communication between the service provider and the identity provider. The artifact itself is an opaque user handle with which the service provider can query the identity provider to receive a full SAML assertion. The motivation for this approach is that the artifact can be small enough in its URI-encoded form to fit in a URI without concern for size limitations. The artifact has the property of being an opaque, pseudo-random nonce that can be used only once. These properties are countermeasures against replay attacks. The randomness property protects the artifact from being guessed by an adversary.

4.4.3.2. Liberty Browser POST Profile

Modern browsers that support JavaScript or ECMAscript can perform the redirect by sending an HTML page with form elements that contain data with a JavaScript or ECMAscript that automatically posts the form. Legacy browsers, or browsers with scripting disabled, must embed the data within the URI.

Note:

The Liberty browser POST profile embeds an assertion within an HTTP form per the form-POST-based redirection (see Section 4.1.2). As a result, this profile does not require any direct communication between the service provider and the identity provider to obtain an assertion. An entire authentication assertion can be included in the posted HTML form because the size allowances for HTML forms are great enough to accommodate one.. See Figure 19.

Figure 19. Example of JavaScript-based HTML form autosubmission with hidden fields

```
<HTML>
<BODY ONLOAD="javascript:document.forms[0].submit()">
<FORM METHOD="POST" ACTION="www.foobar.com/auth">
<INPUT TYPE="HIDDEN" NAME="FOO" VALUE="1234"/>
</FORM>
</BODY>
</HTML>
```

TECHNICAL NOTE:

It must be stressed that Liberty browser POST profile should be supported only in addition to Liberty browser artifact profile due to its dependence on JavaScript (or ECMAscript).

POLICY/SECURITY NOTE:

Implementors and deployers should provide for logging appropriate portions of the authentication assertion.

4.4.3.3. Liberty-Enabled Client and Proxy Profile

The Liberty-enabled client and proxy profile specifies interactions between Liberty-enabled clients and/or proxies, service providers, and identity providers. A Liberty-enabled client is a client that has, or knows how to obtain, knowledge about the identity provider that the user wishes to use with the service provider. In addition a Liberty-enabled client receives and sends Liberty messages in the body of HTTP requests and responses using POST, rather than relying upon HTTP redirects and encoding protocol parameters into URLs. Therefore, Liberty-enabled clients have no restrictions on the size of the Liberty protocol messages.

A Liberty-enabled proxy is a HTTP proxy (typically a WAP gateway) that emulates a Liberty-enabled client.

TECHNICAL NOTE:

The differences between this profile and the other Liberty POST-based profiles are that:

- It does not rely upon HTTP redirects.
- The interactions between the user agent and the identity provider are SOAP-based.
- The Liberty-enabled client and proxy profile includes Liberty-specified HTTP headers in the protocol messages it sends, signifying to identity providers and service providers that it is Liberty-enabled and thus can support capabilities beyond those supported by common non-Liberty-enabled user agents.

4.4.3.4. Single Sign-On Protocol Flow Example: Liberty Artifact Profile

The first step in the single sign-on process in a Liberty artifact profile is that the user goes to a service provider and chooses to log in via the user's preferred identity provider. This login is accomplished by selecting the preferred identity provider from a list presented on the service provider's login page.

TECHNICAL NOTE:

The service provider may discover the preferred identity provider via the identity provider introduction mechanism discussed in Section 4.5 or, in the case of a Liberty-enabled client or proxy, by some other implementation-specific and unspecified means.

Once the user selects the identity provider, the user's browser is redirected to the identity provider with an embedded parameter indicating the originating service provider. The user can then log in to the identity provider as the user normally would. See Figure 20.

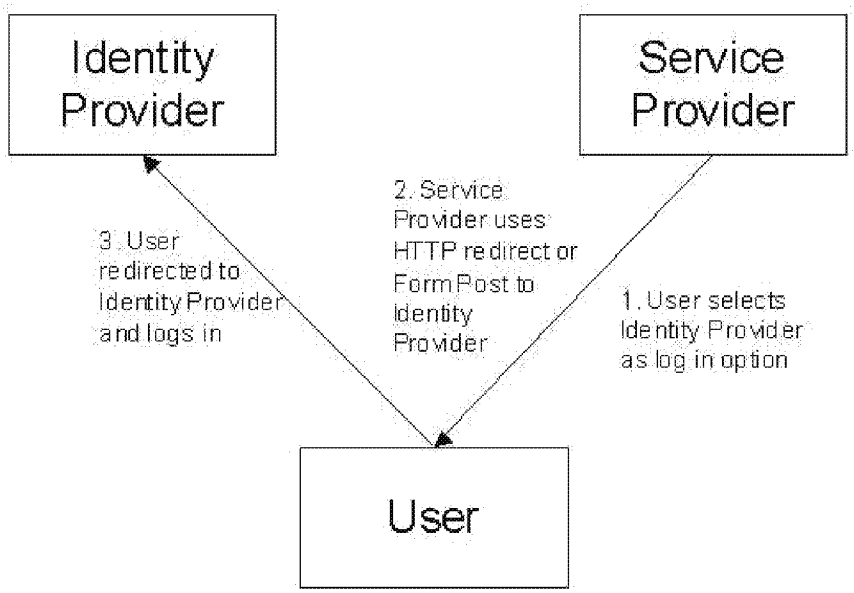
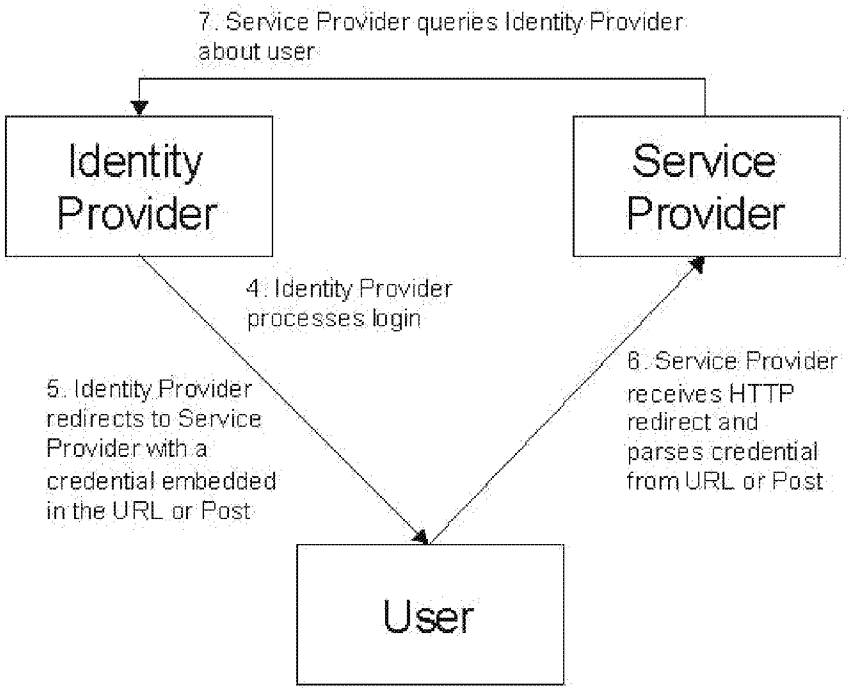


Figure 20. Single sign-on using HTTP redirect / form POST (1 of 2)

The identity provider then processes the login as normal and, upon successful login, redirects the user's browser to the originating service provider with a transient, encrypted credential, called an *artifact*, embedded within the URI. The service provider then parses the artifact from the URI and directly uses it to query the identity provider about the user. In its response, the identity provider vouches for the user, and the service provider may then establish a local notion of session state. See Figure 21.



785

786 **Figure 21. Single sign-on using HTTP redirect / form POST (2 of 2)**

787 4.4.4. Interactions Between Identity Providers

788 In some cases, a Principal may have authenticated with one identity provider, but then be redirected to a second one
789 by a service provider. This may occur either because that service provider has no direct trust relationship with the
790 authenticating identity provider, some previously indicated preference to use the requested identity provider for single
791 sign-on, or the user's direct choice.

792 If the requested identity provider trusts the authenticating identity provider then it may choose to use the Liberty
793 protocols and profiles to initiate a single sign-on request of its own to that provider, the result of which will be used to
794 generate a response to the originally-requesting service provider.

795 In so doing, the user may be relayed between more than one provider during a single sign-on transaction, in order to
796 minimize the need for direct user interaction. An additional consequence is that service providers can be exposed to,
797 but also take advantage of, identity providers that may be outside of their circles of trust. This more strongly models
798 real world interactions between sites, and allows more flexible and convenient user interactions.

799 4.5. Principal Identity Provider Introduction

800 In circle of trusts having more than one identity provider, service providers need a means to discover which identity
801 providers a user is using. Ideally, an identity provider could write a cookie that a service provider could read. However,
802 due to the cookie constraint outlined in Section 4.1.3, an identity provider in one DNS domain has no standardized
803 way to write a cookie that a service provider in another DNS domain can read.

804 A solution to this introduction problem is to use a domain common to the circle of trust in question and thus accessible
805 to all parties, for example, AirlineAffinityGroup.inc or AAG.inc. Entries within this DNS domain will point to IP
806 addresses specified by each affinity group member. For example, service provider CarRental.inc might receive a third-
807 level domain "CarRental.AAG.inc" pointing to an IP address specified by CarRental.inc. The machines hosting this
808 common domain service would be stateless. They would simply read and write cookies based on parameters passed
809 within redirect URLs. This is one of several methods suggested for setting a common cookie in Section 3.6.2 of
810 [LibertyBindProf].

When a user authenticates with an identity provider, the identity provider would redirect the user's browser to the identity provider's instance of a common domain service with a parameter indicating that the user is using that identity provider. The common domain service writes a cookie with that preference and redirects the user's browser back to the identity provider. Then, the user can navigate to a service provider within the circle of trust. See Figure 22.

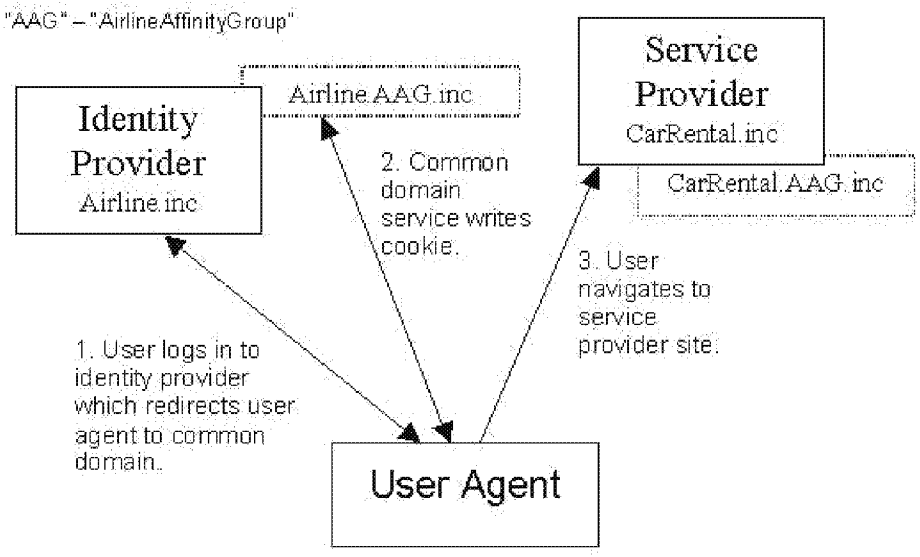


Figure 22. Using a common domain to facilitate introductions (1 of 2)

When the user navigates to a service provider within the circle of trust, the service provider can redirect the user's browser to its instance of the common domain service, which reads the cookie and redirects the user's browser back to the service provider with the user's identity provider embedded in the URL and thus available to service provider systems operating within the service provider's typical DNS domain. See Figure 23.

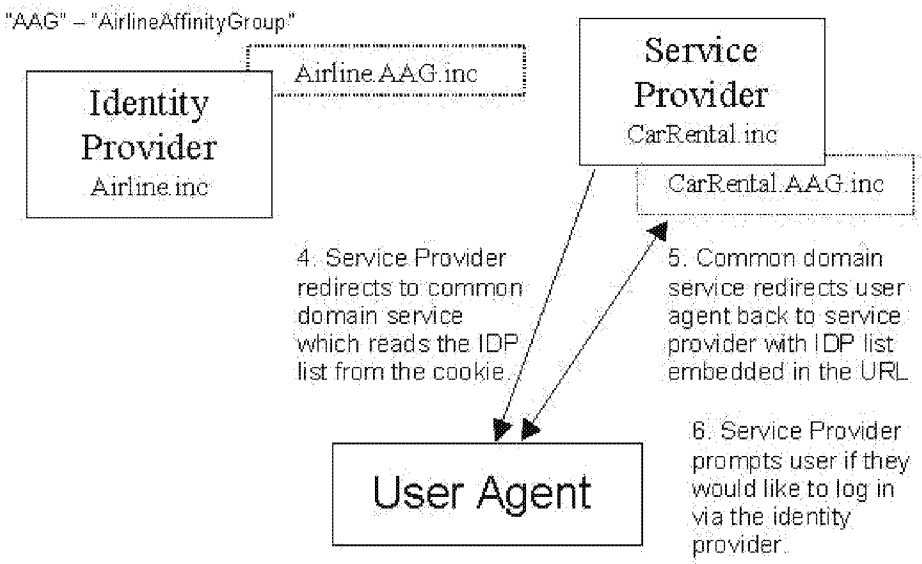


Figure 23. Using a common domain to facilitate introductions (2 of 2)

The service provider now knows with which identity provider the user has authenticated within its circle of trust and can engage in further Liberty protocol operations with that identity provider, for example, single sign-on, on the user's behalf.

POLICY/SECURITY NOTE:

Common Domain Cookie Implications: The identity provider can create either a session common domain cookie (for example, this session only; in practice having ephemeral behavior, see [RFC2965]) or a persistent common domain cookie. The implications with a session cookie are that it will disappear from the user agent cookie cache when the user logs out (although this action would have to be explicitly implemented) or when the user agent is exited. This feature may inconvenience some users. However, whether to use a session or a persistent cookie could be materialized to the user at identity provider login time in the form of a Remember Me checkbox. If not checked, a session cookie is used; if checked, a persistent one is used.

A user security implication of the persistent cookie is that if another person uses the machine, even if the user agent had been exited, the persistent common domain cookie is still present—indeed all persistent cookies are present. See the policy/security note in Section 4.1.3.

However, if the only information contained in a common domain cookie is a list of identity providers—that is, it does not contain any personally identifiable information or authentication information, then the resultant security risk to the user from inadvertent disclosure is low.

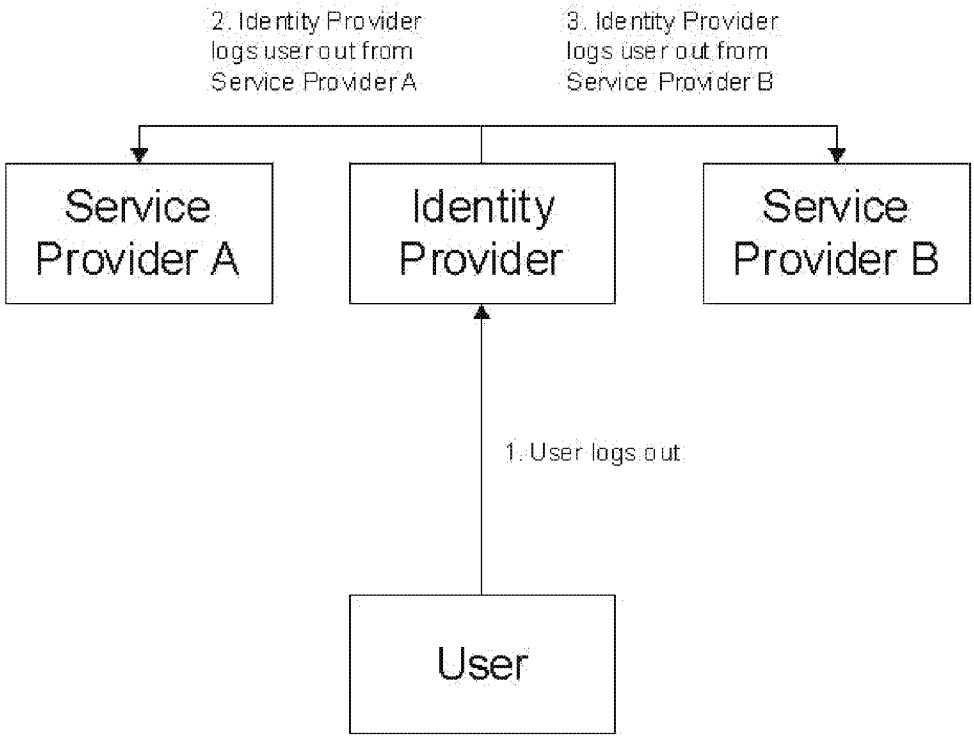
Common Domain Cookie Processing: The manner in which the common domain cookie writing service manipulates the common domain cookie is specified in 3.6.2 of [LibertyBindProf]. The identity provider with which the user most recently authenticated should be the last one in the list of identity providers in the cookie. However, the manner in which service providers interpret the common domain cookie and display choices to the user is unspecified. This lack of specificity implies that service providers may approach it in various ways. One way is to display identity providers in a list ordered in reverse to the order in the common domain cookie. This approach will nominally be in order of most-recently used if the common domain cookie writing service is adhering to the above guideline. Or, the service provider may display only the last identity provider in the list. Or the service provider may display the identity providers in some other order, if needed for some reason(s).

4.6. Single Logout

The Single Logout Protocol and related profiles synchronize session logout functionality across all sessions that were authenticated by a particular identity provider. The single logout can be initiated at either the identity provider (see Figure 24) or the service provider (see Figure 25). In either case, the identity provider will then communicate a logout request to each service provider with which it has established a session for the user.

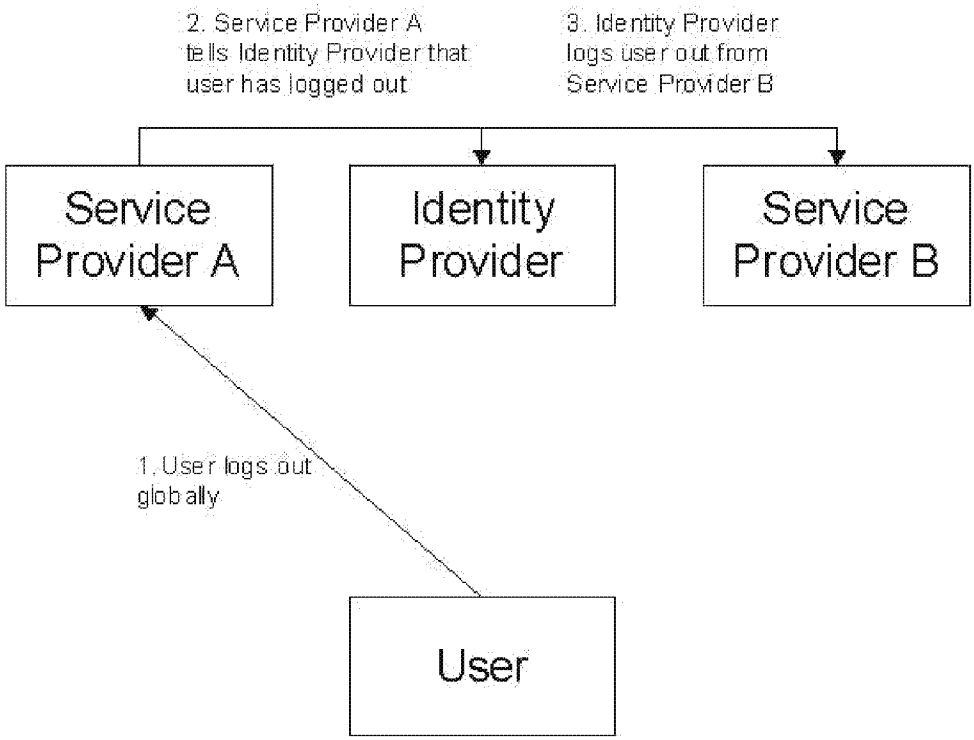
POLICY/SECURITY NOTE:

When using a single sign-on system, it is critical that, when users log out at a service provider, their expectations are set about whether they are logging out from the identity provider or only that particular service provider. It may be necessary to provide both Single Logout and Site Logout buttons or links in Websites so that users' expectations are set. However, site logout may be regarded to come into play only where users have to take a positive action to use their current authentication assertion at a site that they have previously associated with their single sign-on.



868

869 **Figure 24. Single logout from an identity provider**



870

871 **Figure 25. Single logout from a service provider**

4.6.1. Single Logout Profiles

[LibertyBindProf] specifies three overall profiles for communicating the logout request among service providers and an identity provider:

- **HTTP-Redirect-Based:** on using HTTP 302 redirects
- **HTTP-GET-Based:** Relies on using HTTP GET requests of IMG tags
- **SOAP/HTTP-Based:** Relies on SOAP over HTTP messaging

All three profiles may be initiated at an identity provider. Only the first and the last may be initiated at a service provider. See [LibertyBindProf] for details.

TECHNICAL NOTE:

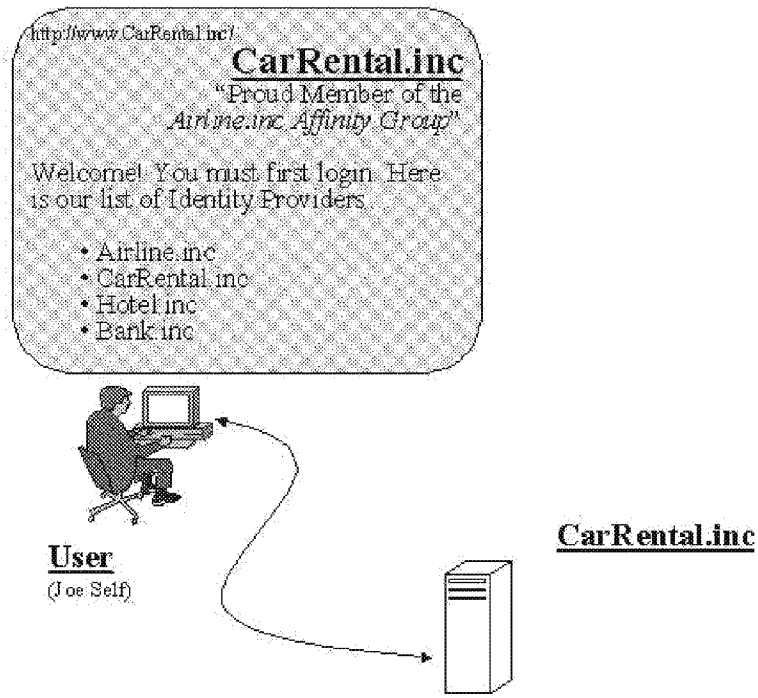
The user-perceivable salient difference between the single logout profiles is that with the HTTP-redirect-based and SOAP/HTTP-based profiles, the Webpage from which the user initiates the logout process will remain in place as the logout process occurs (that is, each service provider is contacted in turn), while with the HTTP-GET-based profile, the identity provider has the opportunity to reload images (one per service provider, for example, completion check marks) on the viewed Webpage as the logout process proceeds.

4.7. Example User Experience Scenarios

This section presents several example user experience scenarios based upon the federation, introduction, and single sign-on facets of the Liberty Version 1.2 architecture. The intent is to illustrate the more subtle aspects of the user experience at login time and to illustrate common Web-specific user interface techniques that may be employed in prompting for, and collecting, the user's credentials. Specific policy and security considerations are called out.

4.7.1. Scenario: Not Logged in Anywhere, No Common Domain Cookie

In this scenario, Joe Self is not logged in at any Website, does not have a common domain cookie (for example, he restarted his user agent and/or flushed the cookie cache), and surfs to CarRental.inc. without first visiting his identity provider, Airline.inc.



895

896 **Figure 26. User arrives at service provider's Website without any authentication evidence or common domain cookie**

897 CarRental.inc presents Joe Self with a welcome page listing identity providers from which he can select (see
898 Figure 26). Joe Self selects Airline.inc from the list.

899 Section 4.7.1.1 through Section 4.7.1.3 illustrate three different, plausible, Web-specific user interface techniques
900 CarRental.inc, working in concert with Airline.inc, may use to facilitate Joe Self's login:

901 • Redirect to identity provider Website

902 • Identity provider dialog box

903 • Embedded form

TECHNICAL NOTE:

These user interface techniques are commonly employed in Web-based systems. They are not particular to, or specified by, Liberty. They are presented for illustrative purposes only.

4.7.1.1. Login via Redirect to Identity Provider Website

With login via redirect to the identity provider's Website, service providers provide direct links, likely effected via redirects, to the identity provider's appropriate login page. Joe Self's browser will display an identity provider's Webpage (see Figure 27); and upon successful login, his browser will be redirected back to the service provider's Website where Joe Self will be provided access (see Figure 30).

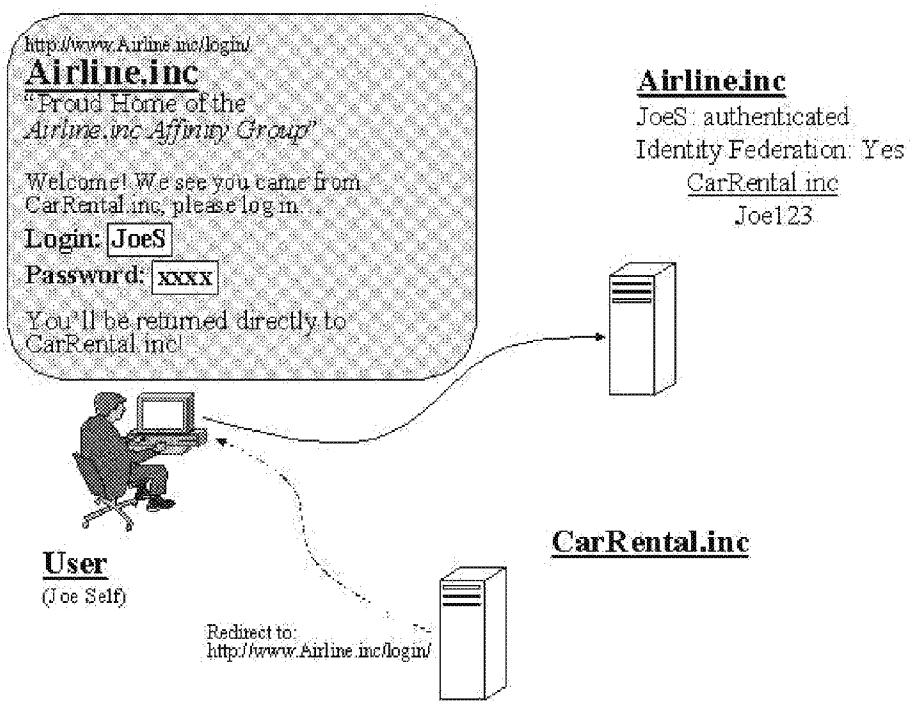


Figure 27. User arrives at service provider's Website without any authentication evidence or common domain cookie

POLICY/SECURITY NOTE:

Service provider redirects to identity provider's login page.

4.7.1.2. Login via Identity Provider Dialog Box

With login via a dialog box from the identity provider, the links on the service provider's Webpage invoke a dialog or popup box. Joe Self's browser will display an identity provider popup (see Figure 28); and upon successful login, the popup box will close, and Joe Self will be provided access at the service provider's Website (see Figure 30).

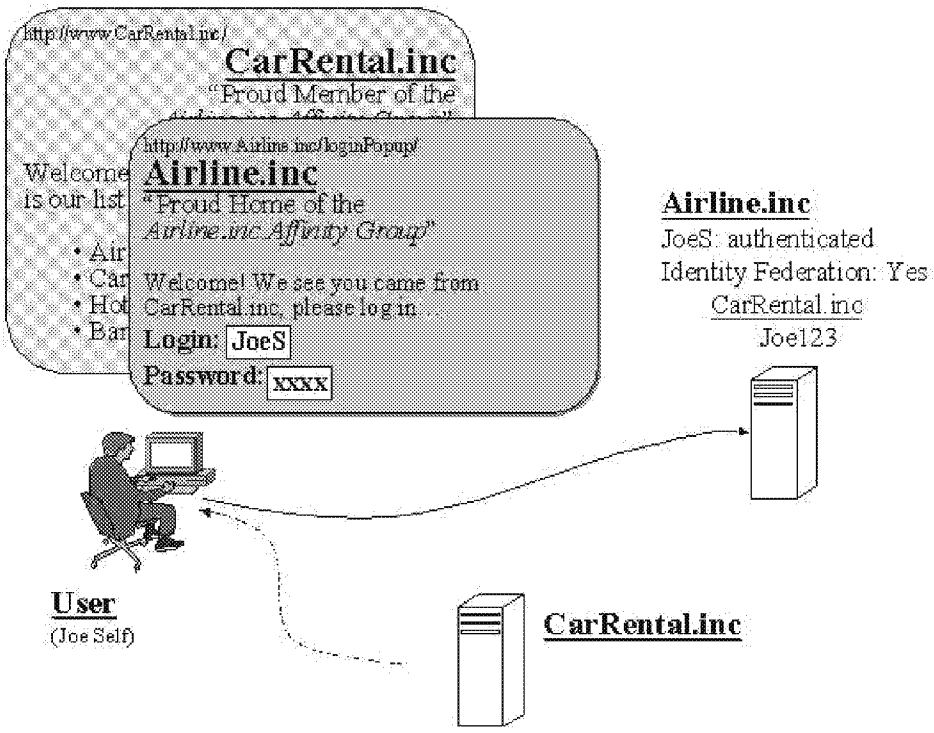


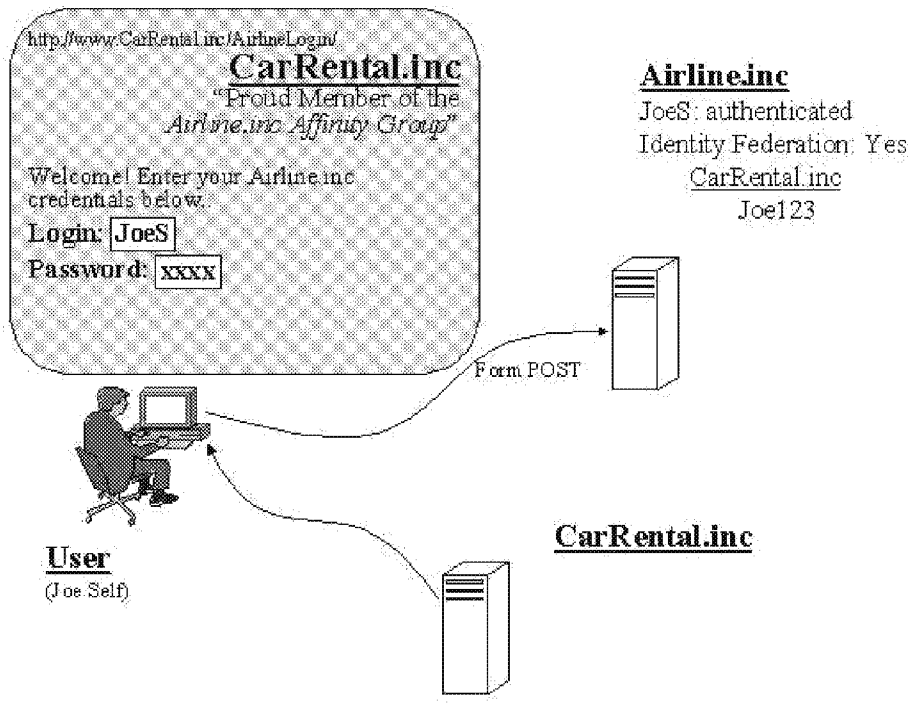
Figure 28. Service provider invokes dialog or popup box from identity provider.

POLICY/SECURITY NOTE:

Login via a dialog box from the identity provider is relatively secure in that the user reveals his credentials directly to the identity provider. Of course, the usual security considerations surrounding login and authentication events apply.

4.7.1.3. Login via Embedded Form

With login via embedded form, the links on the service provider's Webpage cause the service provider to display embedded login forms. In other words, the displayed page comes from the service provider, but when Joe Self presses the Submit button, the information is conveyed to the identity provider, typically via POST (see Figure 20). To Joe Self, it appears as if he has not left the service provider's Webpages. Upon successful login, Joe Self will be provided access at the service provider's Website (see Figure 30).



932

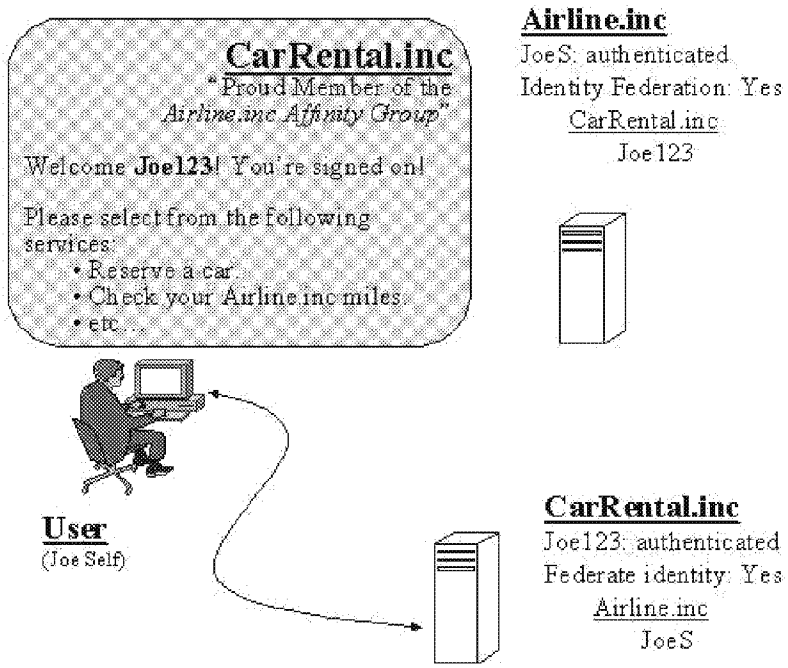
933 **Figure 29. Login via embedded form**

934 **POLICY/SECURITY NOTE:**

935 Although users may like the seamlessness of this embedded form mechanism and deployers will like that the
936 user does not leave their Website, it has serious policy and security considerations. In this mechanism, the
937 user may be revealing his identity provider credentials to the service provider in cleartext. This is because
938 the service provider controls the actual code implementing both the page and the embedded form and thus
939 can conceivably capture users' credentials. In this way, privacy surrounding the user's identity provider
940 account may be compromised by such a rogue service provider, who could then wield those credentials and
941 impersonate the user. Because of this, when using authentication via embedded form, deployers may want to
942 consider appropriate contract terms between identity providers and service providers to address this risk.

943 ### 4.7.1.4. The User is Logged in at CarRental.inc

944 CarRental.inc and Airline.inc then work in conjunction to effect login, and the CarRental.inc Website establishes a
945 session based upon Joe Self's identity federation with Airline.inc (see Figure 30).



946

947 **Figure 30. Login via embedded form**

948 4.7.2. Scenario: Not Logged in Anywhere, Has a Common Domain Cookie

949 This scenario is similar the prior one. The only difference is that Joe Self's browser already has a common domain
950 cookie cached. Therefore, when he arrives at a CarRental.inc Webpage, CarRental.inc will immediately know with
951 which identity provider Joe Self is affiliated (Airline.inc in this case). It can immediately perform login via one of the
952 three mechanisms outlined in the prior example or may prompt the user first.

953 **POLICY/SECURITY NOTE:**

954 Implementors and deployers should make allowance for the user to decide whether to immediately authen-
955 ticate with the identity provider or be offered the chance to decline and authenticate either locally with the
956 service provider or select from the service provider's list of affiliated identity providers.

957 4.7.3. Scenario: Logged in, Has a Common Domain Cookie

958 This scenario is illustrated in 2.2.

References

Informative

[LibertyBindProf] Cantor, Scott, Kemp, John, Champagne, Darryl, eds. "Liberty ID-FF Bindings and Profiles Specification," Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004). *http://www.projectliberty.org/specs*

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version 1.2-errata-v3.0, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyAuthnContext] Madsen, Paul, eds. "Liberty ID-FF Authentication Context Specification," Version 1.3, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 1.1, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyGlossary] "Liberty Technical Glossary," Version 1.4, Liberty Alliance Project (14 Dec 2004). *http://www.projectliberty.org/specs* Hodges, Jeff, eds.

[LibertyImplGuide] "Liberty ID-FF Implementation Guidelines," Version 1.2, Liberty Alliance Project (18 April 2004). *http://www.projectliberty.org/specs* Thompson, Peter, Champagne, Darryl, eds.

[SAMLBind] Mishra, Prateek, eds. (05 November 2002). "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)," Version 1.0, OASIS Standard, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/security/#documents*

[RFC1738] Berners-Lee, T., Masinter, L., McCahill, M., eds. (December 1994). "Uniform Resource Locators (URL)," RFC 1738., Internet Engineering Task Force *http://www.ietf.org/rfc/rfc1738.txt [December 1994].*

[RFC2119] Bradner, S., eds. "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet Engineering Task Force (March 1997). *http://www.ietf.org/rfc/rfc2119.txt [March 1997].*

[RFC2965] Kristol, D., Montulli, L., eds. (October 2000). "HTTP State Management Mechanism," RFC 2965., Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2965.txt [October 2000].*

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June 1999). "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2616.txt [June 1999].*

[RFC2396] Berners-Lee, T., Fielding, R., Masinter, L., eds. (August 1998). "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2396.txt [August 1998].*

[SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David , Kakivaya, Gopal, Layman, Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C Note (08 May 2000). *http://www.w3.org/TR/2000/NOTE-SOAP-20000508/*

EXHIBIT 4

Liberty ID-FF Bindings and Profiles Specification

Version: 1.2-errata-v2.0

Editors:
Scott Cantor, Internet2, The Ohio State University
John Kemp, IEEE-ISTO
Darryl Champagne, IEEE-ISTO

Contributors:
Robert Aarts, Nokia Corporation
Bronislav Kavsan, RSA Security Inc.
Thomas Wason, IEEE-ISTO

Abstract:

Specification of the Liberty Alliance Project core profiles and bindings. This specification defines the bindings and profiles of the Liberty protocols and messages to HTTP-based communication frameworks. This specification relies on the SAML core framework in SAML Core V1.1 and makes use of adaptations of the SAML profiles in SAML Bindings V1.1.

Filename: draft-liberty-idff-bindings-profiles-1.2-errata-v2.0.pdf

Notice

This document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the document solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this document for other uses must contact the Liberty Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this document may require licenses under third party intellectual property rights, including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are not, and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. **This Specification is provided "AS IS", and no participant in the Liberty Alliance makes any warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringement of third party intellectual property rights, and fitness for a particular purpose.** Implementors of this Specification are advised to review the Liberty Alliance Project's website (http://www.projectliberty.org/) for information concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance Management Board.

Liberty Alliance Project
Licensing Administrator
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08855-1331, USA
info@projectliberty.org

Contents

1. Introduction

This specification defines the bindings and profiles of the Liberty protocols and messages to HTTP-based communication frameworks. This specification relies on the SAML core framework in [SAMLCore11] and makes use of adaptations of the SAML profiles in [SAMLBind11]. A separate specification, [LibertyProtSchema], is used to define the Liberty protocols and messages used within the profiles. Definitions for Liberty-specific terms can be found in [LibertyGlossary].

1.1. Notation

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this specification are to be interpreted as described in [RFC2119]: "they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)."

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

```
Listings of productions or other normative code appear like this.
```

```
Example code listings appear like this.
```

Note:

Non-normative notes and explanations appear like this.

Conventional XML namespace prefixes are used throughout this specification to stand for their respective namespaces as follows, regardless of whether a namespace declaration is present in the example:

XML Namespace Conventions

- The prefix `lib:` stands for the Liberty namespace `urn:liberty:iff:2003-08`
- The prefix `saml:` stands for the SAML assertion namespace (see [SAMLCore]).
- The prefix `samlp:` stands for the SAML request-response protocol namespace (see [SAMLCore]).
- The prefix `ds:` stands for the W3C XML signature namespace, `http://www.w3.org/2000/09/xmldsig#`
- The prefix `xenc:` stands for the W3C XML encryption namespace, `http://www.w3.org/2001/04/xmlenc#`
- The prefix `SOAP-ENV:` stands for the SOAP 1.1 namespace, `http://schemas.xmlsoap.org/soap/envelope` (see [SOAP1.1]).

Terminology from [RFC2396] is used to describe components of an HTTP URL. An HTTP URL has the following form:

```
<scheme>://<authority><path>?<query>
```

Sections in this document specify certain portions of the `<query>` component of the URL. Ellipses (...) are used to indicate additional, but unspecified, portions of the `<query>` component.

2. Protocol Bindings

The Liberty protocol bindings are defined in this section.

2.1. SOAP Binding for Liberty

The Liberty SOAP binding defines how to use SOAP to send and receive Liberty protocol requests and responses using SOAP 1.1 messages.

Like Liberty, SOAP can be used over multiple underlying transports. This binding has protocol-independent aspects, but REQUIRES the use of SOAP over HTTP.

2.1.1. Protocol-Independent Aspects of the Liberty SOAP Binding

The following sections define aspects of the Liberty SOAP binding that are independent of the underlying protocol, such as HTTP, on which the SOAP messages are transported.

2.1.1.1. Basic Operation

SOAP messages consist of three elements: an envelope, header data, and a message body. Liberty request-response protocol elements MUST be enclosed within the SOAP message body.

SOAP 1.1 also defines an optional data encoding system. This system is not used within the Liberty SOAP binding. This means that SAML messages can be transported using SOAP without re-encoding from the "standard" Liberty schemas to one based on the SOAP encoding.

The specific profile determines the type of messages that can be sent or received. The system model used for Liberty conversations over SOAP may be a simple request-response model, or it may be a more complex interaction that includes HTML forms or other input mechanisms that interact with a Principal.

This Liberty specification defines constraints. Liberty protocol messages MUST be sent as the top level element in the SOAP body. The requester or responder MUST NOT include more than one Liberty protocol message in a single SOAP message. The requester or responder MUST NOT include any additional XML elements in the SOAP body. Additionally, if a SOAP fault code is returned, then no Liberty protocol message may appear in the SOAP body. SOAP faults MUST only be used for signaling non-Liberty-related errors.

[SOAPv1.1] references an early draft of the XML Schema specification including an obsolete namespace. Originators of Liberty SOAP messages SHOULD generate SOAP messages referencing only the final XML schema namespace. Receivers of Liberty SOAP messages MUST be able to process both the XML schema namespace used in [SOAPv1.1] and the final XML schema namespace.

2.1.1.2. SOAP Headers

A Liberty SOAP message MAY contain arbitrary headers added to the SOAP message. This binding does not define any additional SOAP headers.

Liberty SOAP messages MUST NOT require that any headers be understood for correct interpretation of the message.

2.1.1.3. Authentication

Authentication of Liberty messages is OPTIONAL and depends on the environment of use. Authentication protocols available from the underlying substrate protocol MAY be utilized to provide authentication. Section 2.1.2.1 describes authentication in the SOAP-over-HTTP environment.

2.1.1.4. Message Integrity

Message integrity of Liberty messages is OPTIONAL and depends on the environment of use. The security layer in the underlying substrate protocol MAY be used to ensure message integrity. Section 2.1.2.2 describes support for message integrity in the SOAP-over-HTTP environment.

2.1.1.5. Confidentiality

Confidentiality of Liberty messages is OPTIONAL and depends on the environment of use. The security layer in the underlying substrate protocol MAY be used to ensure message confidentiality. Section 2.1.2.3 describes support for confidentiality in the SOAP over HTTP environment.

2.1.2. Use of SOAP over HTTP

This section describes certain specifics of using SOAP over HTTP, including HTTP headers, error reporting, authentication, message integrity and confidentiality.

The HTTP binding for SOAP is described in [SOAPv1.1] §6.0. It requires the use of a SOAPAction header as part of a SOAP HTTP request. Processing of a Liberty message MUST NOT depend on the value of this header. A Liberty message MAY set the value of its SOAPAction header as follows:

```
urn:liberty:soap-action
```

2.1.2.1. Authentication

Liberty SOAP message endpoints MUST implement the following authentication methods:

1. No client or server authentication.
2. HTTP basic client authentication [RFC2617] with and without SSL 3.0 or TLS 1.0.
3. HTTP over SSL 3.0 or TLS 1.0 (see Section 6) server authentication with a server-side certificate.
4. HTTP over SSL 3.0 or TLS 1.0 client authentication with a client-side certificate.

If a message receiver uses SSL 3.0 or TLS 1.0, it MUST use a server-side certificate.

2.1.2.2. Message Integrity

When message integrity needs to be guaranteed, messages MUST be sent with HTTP over SSL 3.0 or TLS1.0 with a server-side certificate.

2.1.2.3. Message Confidentiality

When message confidentiality is required, messages MUST be sent with HTTP over SSL 3.0 or TLS 1.0 with a server-side certificate.

2.1.2.4. Security Considerations

Before deployment in a given profile, each combination of authentication, message integrity and confidentiality mechanisms SHOULD be analyzed for vulnerability in the context of the profile.

[RFC2617] describes possible attacks in the HTTP environment when basic or message-digest authentication schemes are used.

2.1.2.5. Error Reporting

A message receiver that refuses to perform a message exchange SHOULD return a "403 Forbidden" response. In this case, the content of the HTTP body is not significant.

As described in [SOAPv1.1] § 6.2, in the case of a SOAP error while processing a SOAP request, the SOAP HTTP server MUST return a "500 Internal Server Error" response and include a SOAP message in the response with a SOAP fault element. This type of error SHOULD be returned for SOAP-related errors detected before control is passed to the Liberty message processor, or when the SOAP processor reports an internal error (for example, the SOAP XML namespace is incorrect).

In the case of a Liberty processing error, the SOAP HTTP server MUST respond with "200 OK" and include a profile-specified response as the only child of the <SOAP-ENV:Body> element.

2.1.2.6. Example of Message Exchange Using SOAP over HTTP

The following is an example of the SOAP exchange for the single sign-on browser artifact profile requesting an authentication assertion (the left margin white space added for legibility invalidates the signature).

```
POST /authn HTTP/1.1
Host: idp.example.com
Content-type: text/xml
Content-Length: nnnn
<soap-env:Envelope
   xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
   <soap-env:Header/>
   <soap-env:Body>
      <samlp:Request xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
         xmlns:lib="urn:liberty:iff:2003-08"
         xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
         xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         IssueInstant="2002-12-12T10:08:56Z"
         MajorVersion="1"
         MinorVersion="1"
         RequestID="e4d71c43-c89a-426b-853e-a2b0c14a5ed8"
         id="ericssonb6dc3636-f2ad-42d1-9427-220f2cf70ec1">
         <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:SignedInfo>
               <ds:CanonicalizationMethod
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
               </ds:CanonicalizationMethod>
               <ds:SignatureMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">
               </ds:SignatureMethod>
               <ds:Reference URI="#ericssonb6dc3636-f2ad-42d1-9427-220f2cf70ec1">
                  <ds:Transforms>
                     <ds:Transform
                        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature">
                     </ds:Transform>
                     <ds:Transform
                        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                     </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
                     Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
                  </ds:DigestMethod>
                  <ds:DigestValue>+k6Tno Gk PKZ pUQVyokFdwkcE=</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>
               wXJMVcPOlV1jFnWJPyOWqPbGqm8A1+/2bbgNzF4L4LMu4yEcRtttLdPPT3bvbwkwllXjL9Nn0F
               bYEyiVzlNcjAxX0LfgwutvEdJb/48IU4L 8obXPXfqTZLiBX1RbICRmRvj1PIu22oGCV6Ewu iWRv
               OD6Ox9svtSgFJ iXkZQ
            </ds:SignatureValue>
```

```
            <ds:KeyInfo>
               <ds:X509Data>
                  <ds:X509Certificate>
                     MIIDMTCCApqgAwIBAgIBADANBgkqhkiG9w0BAQQFADCBlTELMAkGA1UEBhMCVVMxCzAJBgNVBAcT
                     AlNGMRkwFwYDVQQKExBMaWJlcnR5IEFsbGlhbmNlMRQwEgYDVQQLEwtJT1AgVGVzdGVyczEiMCAG
                     A1UEAxM2TGliZXJ0eSBUZXN0ZXJzIENlcnRpZmljYXRlczEXMCIGCSqGSIb3DQEJARYVcnJvZHJp23Vl
                     ekBuZW9zo2wuomV0MB4XDTAyMTIwNJE1NTg0NFoXDTEyMTIwMTE1NTg0NFowgasxCzAJBgNVBAYT
                     AlVTMQswCQYDVQQHEwJTRjEkMCIGA1UEChMbTGliZXJ0eSBBbGxpYW5jZSBlcmljc3Nvbi1hMSYw
                     JAYDVQQLEx1JT1AgVGVzdGVycyBlcmljc3Nvbi1hIHNpZ25lcjEXMBUGA1UEAxMOZXJpY3Nzb24t
                     YS5pb3AxKDAmBgkqhkiG9w0BCQEWGXJyb2RyaWd1ZXpAZXJpY3Nzb24tYS5pb3AwqZ8wDQYJKoZI
                     hvcNAQEBBQADqY0AMIGJAoGBAPUoGYvJxQc5jzDnJ14TV6TaTbB3fH95ju24Z0y6HQxm6qXdJSAo
                     Wh7/AIes4JcV09DC2kZS6Vow2YoXt2LIyH9HWH2zEUz1jS/PUeBHEWcW3tFezM6ch5GG5rCuVPZa
                     W9eoGJoFPSzOPFKUAwdHUXSDWufY1KZ93IxhOBeZqq6VAqMBAAGjeTB3MEoGCWCGSAGG+EIBDQQ9
                     FjtUaGlzIHNpZ25pbmcgY2VydCB3YXMgY3JlYXRlZCBmb3IgdGVzdGluZy4gRG8gbm90IHRydXN0
                     IG10LjAJBgNVHRMEAjAAMB3GCWCGSAGG+EIBAQQEAwIEMDALBgNVHQ8EBAMCBsAwDQYJKoZIhvcN
                     AQEEBQADgYEAR/HSg3pAprQwQVyWDE9pCaicuKv4/W/+hrdpXlVKSr6TIlg4ouDCQJNos7tNuG9Z
                     AbfWLHvCss51N2cfAzfns/JXqxRqcsxzL5ZUBksPpmsDoboopUv6Xm8RFsi7yB9AGaVuqCbeY/+m
                     70nOu03O+FlMN3U1k2E3rOXXlU1roCC
                  </ds:X509Certificate>
               </ds:X509Data>
            </ds:KeyInfo>
         </ds:Signature>
         <samlp:AssertionArtifact>
            AAM1uXw6+f+jyA/4XuFHqPl7QJvd/LIQL9+t7YQtG1GWk9bph0Adl+o+
         </samlp:AssertionArtifact>
      </samlp:Request>
   </soap-env:Body>
</soap-env:Envelope>
```

The following is an example of a response, which supplies an assertion containing an authentication statement:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn
<soap-env:Envelope
   xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
   <soap-env:Header/>
   <soap-env:Body>
      <samlp:Response
         xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
         InResponseTo="RPCUk2ll+GVz+UllURp5loFvJXk"
         IssueInstant="2002-10-31T21:42:13Z" MajorVersion="1" MinorVersion="1"
         Recipient="http://localhost:8080/sp"
         ResponseID="LANWfL2xLybnc+BCwgY+p1/vIVAj">
         <samlp:Status>
            <samlp:StatusCode
               xmlns:qns="urn:oasis:names:tc:SAML:1.0:protocol"
               Value="qns:Success">
            </samlp:StatusCode>
         </samlp:Status>
         <saml:Assertion
            xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
            xmlns:lib="urn:liberty:iff:2003-08"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            AssertionID="SqMC8Hs2vJ7Z+t4UiLSmhKOSU00U"
            InResponseTo="RPCUk2ll+GVz+UllURp5loFvJXk"
            IssueInstant="2002-10-31T21:42:13Z" Issuer="http://localhost:8020/idp"
            MajorVersion="1" MinorVersion="2"
            xsi:type="lib:AssertionType">
            <saml:Conditions
               NotBefore="2002-10-31T21:42:12Z"
               NotOnOrAfter="2002-10-31T21:42:43Z">
               <saml:AudienceRestrictionCondition>
                  <saml:Audience>http://localhost:8080/sp</saml:Audience>
               </saml:AudienceRestrictionCondition>
```

```
            </saml:Conditions>
            <saml:AuthenticationStatement
               AuthenticationInstant="2002-10-31T21:42:13Z"
               AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
               xsi:type="lib:AuthenticationStatementType">
               <saml:Subject xsi:type="lib:SubjectType">
                  <saml:NameIdentifier Format="urn:liberty:iff:nameid:federated">
                   C9FfGouQdBJ7bpkismYgd8ygeVb3P1WK
                  </saml:NameIdentifier>
                  <saml:SubjectConfirmation>
                     <saml:ConfirmationMethod>
                        urn:oasis:names:tc:SAML:1.0:cm:artifact
                     </saml:ConfirmationMethod>
                  </saml:SubjectConfirmation>
                  <lib:IDPProvidedNameIdentifier>
                     C9FfGouQdBJ7bpkismYgd8ygeVb3P1WK
                  </lib:IDPProvidedNameIdentifier>
               </saml:Subject>
            </saml:AuthenticationStatement>
            <ds:Signature>
               <ds:SignedInfo>
                  <ds:CanonicalizationMethod
                     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                  </ds:CanonicalizationMethod>
                  <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">
                  </ds:SignatureMethod>
                  <ds:Reference URI="">
                     <ds:Transforms>
                        <ds:Transform
                           Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature">
                        </ds:Transform>
                        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        </ds:Transform>
                     </ds:Transforms>
                     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
                     </ds:DigestMethod>
                     <ds:DigestValue>ZbscbqHTX9H8bBftRTWlG4EgvlA=</ds:DigestValue>
                  </ds:Reference>
               </ds:SignedInfo>
               <ds:SignatureValue>
                  H-q3nC3jUaljluKUVkcC4iTFC1xeZQIFF0nvIlqPS5oZhtkBaDb9qITA7gIkotaB584wXqTXwsfsu
                  IrwTbuL3r85RjVIF6NeCeiy3K0+z3uewxyeZPz8wna449VNm0qNIlYkgNak9ViNCp0/ks5MAttoPo
                  2fLOfaKu3wWG6d1GIDM=
               </ds:SignatureValue>
            </ds:Signature>
         </saml:Assertion>
      </samlp:Response>
   </soap-env:Body>
</soap-env:Envelope>
```

2.1.2.7. Example of Message Exchange Using URL-encoding

```
http://127.0.0.1:8080/tfsidp/IDPSingleSignOnServiceV12?
   RequestID=ddd4aa20-24d4-4366-8f6C-7bbCeC68334a&MajorVersion=1
   &MinorVersion=2&IssueInstant=2003-07-26T05%3A44%3A00Z
   &ProviderID=http%3A%2F%2F127.0.0.1%3A8081%2Ftfssp
   &NameIDPolicy=none&IsPassive=0
   &RelayState=3f53f0934e63726de06f555493683e131ad131ff
   &SigAlg=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1
   &Signature=
      XAfiiHlyI8OJG6IGV6Y17ToaxwF1E4wv%2FrONqAkkCllKyxvjcO3%2Bitx7Q44v
      RVDm%2BbAm38ofFSsb%0AbzYsWgTMAiDPsm2wQUuYP2oRB9AAoS%2BdZwfGWhqV%
      2FU0malW7cRMDBA2ewyAQTdiALLesetQ6zAnJ%0A4KzHExQ1%2BzelPehsWOE%3D
```

330 **Example 1. URL-encoded <AuthnRequest>**

331
332 http://127.0.0.1:8081/tfssp/SPAssertionConsumerServiceV_2?
333 SAMLart=AAMZf3s1j11R%23gQe%23MVfP6flshoGaPG1XrZG5E%2B212nHDCxPA7h%2BDcE%2B
334 &RelayState=3f53f0934e63726de06f555493683e131ad131ff
335

336 **Example 2. URL-encoded <AuthnResponse>**

3. Profiles

This section defines the Liberty profiles for the use of request and response messages defined in [LibertyProtSchema] and [SAMLCore11]. The combination of message content specification and message transport mechanisms for a single client type (that is, user agent) is termed a *Liberty profile*. The profiles have been grouped into categories according to the protocol message intent.

The following profile categories are defined in this document:

- **Single Sign-On and Federation:** The profiles by which a service provider obtains an authentication assertion from an identity provider facilitating single sign-on and identity federation.
- **Name Registration:** The profiles by which service providers and identity providers specify the name identifier to be used when communicating with each other about the Principal.
- **Federation Termination Notification:** The profiles by which service providers and identity providers are notified of federation termination.
- **Single Logout:** The profiles by which service providers and identity providers are notified of authenticated session termination.
- **Identity Provider Introduction:** The profile by which a service provider discovers which identity providers a Principal may be using.
- **Name Identifier Mapping:** The profile by which a service provider may obtain a NameIdentifier with which to refer to a Principal at a SAML Authority.
- **Name Identifier Encryption:** The profile by which one provider may encrypt a NameIdentifier to permit it to pass through a third-party without revealing the actual value until received by the intended provider.

3.1. Common Requirements

The following rules apply to all profiles in this specification, unless otherwise noted by the individual profile.

1. All HTTP requests and responses MUST be drawn from either HTTP 1.1 (see [RFC2616]) or HTTP 1.0 (see [RFC1945]). When an HTTP redirect is specified, the HTTP response MUST have a status code of "302.". According to HTTP 1.1 and HTTP 1.0, the use of status code 302 is recommended to indicate "the requested resource resides temporarily under a different URI." The response may also include additional headers and an optional message.
2. When `https` is specified as the `<scheme>` for a URL, the HTTP connection MUST be made over either SSL 3.0 (see [SSL]) or TLS 1.0 (see [RFC2246]) or any subsequent protocols that are backwards compatible with SSL 3.0 and/or TLS 1.0. Other security protocols MAY be used as long as they implement equivalent security measures.
3. Messages between providers MUST have their integrity protected, confidentiality MUST be ensured and the recipient MUST authenticate the sender.
4. Providers MUST use secure transport (`https`) to achieve confidentiality and integrity protection. The initiator of the secure connection MUST authenticate the server using server-side X.509 certificates.

5. The authenticated identity of an identity provider MUST be securely available to a Principal before the Principal presents his/her personal authentication data to that identity provider.

6. Certificates and private keys MUST be suitable for long-term signatures. See [LibertyProtSchema] for guidelines on signature verification. For signing and verification of protocol messages, identity and service providers SHOULD use certificates and private keys that are distinct from the certificates and private keys applied for SSL or TLS channel protection.

7. In transactions between service providers and identity providers, requests MUST be protected against replay, and received responses MUST be checked for correct correspondence with issued requests. (**Note:** Other steps may intervene between the issuance of a request and its eventual response within a multistep transaction involving redirections.) Additionally, time-based assurance of freshness MAY be provided.

8. Each service provider within a circle of trust MUST be configured to enable identification of the identity providers whose authentications it will accept. Each identity provider MUST be configured to enable identification of the service providers it intends to serve.
Note:
The format of this configuration is a local matter and could, for example, be represented as lists of names or as sets of X.509 certificates of other circle of trust members.

9. Circle of trust bilateral agreements on selecting certificate authorities, obtaining X.509 credentials, establishing and managing trusted public keys, and tracking lifecycles of corresponding credentials are assumed and not in scope for this specification.

10. The `<scheme>` of the URL for SOAP endpoints MUST be `https`.

11. All SOAP message exchanges MUST adhere to the SOAP protocol binding for Liberty (see Section 2.1).

3.1.1. User Agent

Unless otherwise noted in the specific profile, a user agent MUST support the following features to be interoperable with the protocols in [LibertyProtSchema] and Liberty profiles in this document:

- HTTP 1.0 (see [RFC1945]) or HTTP 1.1 (see [RFC2616]).
- SSL 3.0 (see [SSL]) or TLS 1.0 (see [RFC2246]) or any subsequent protocols which are backwards compatible with SSL 3.0 and/or TLS 1.0 either directly or via a proxy (for example, a WAP gateway).
- Minimum maximum URL length of 256 bytes. See [LibertyGlossary] for definition.
- A WAP browser user agent MUST support WML 1.0,1.1, 1.2 or 1.3 [WML] in addition to the above requirements.

Additionally, to support the optional identity provider introduction profile, either the user agent or a proxy must support session cookies (see [RFC2965]). The issue of using persistent cookies or session-length cookies is discussed in [LibertyImplGuide].

3.1.2. Formatting and Encoding of Protocol Messages

All protocol messages that are indicated by the profile as being communicated in the `<query>` component of the URL MUST adhere to the formatting and encoding rules in Section 3.1.2.1.

3.1.2.1. Encoding URL-embedded Messages

URL-embedded messages are encoded using the `application/x-www-form-urlencoded` MIME type as if they were generated from HTML forms with the GET method as defined in [HTML4].

The original XML protocol message MUST be encoded as follows:

- The `<query>` component parameter value MUST be the value of the XML protocol message element or attribute value.
- The value of the `<query>` component parameter MUST be a space-delimited list when the original message element has multiple values.
- Some of the referenced protocol message elements and attributes are optional. If an optional element or attribute does not appear in the original XML protocol message, then the corresponding data item MUST be omitted from the URL encoded message.
- URLs appearing in the URL-encoded message SHOULD NOT exceed 80 bytes in length (including %-escaping overhead). Likewise, the `<lib:RelayState>` data value SHOULD NOT exceed 80 bytes in length.
- The URL-encoding of status codes in the responses `RegisterNameIdentifierResponse` and `LogoutResponse` may be taken from several sources. The top level codes MUST be from SAML. Other codes (including Liberty-defined values) MAY be used at the second or lower levels. The URL parameter value should be interpreted as a QName with the "`lib`", "`saml`", and "`samlp`" namespaces pre-defined to their respective namespace URIs. Query parameters with the name "`xmlns:prefix`" can be used to map additional namespace prefixes for the purpose of QName resolution, so long as the `xmlns:prefix` URL parameter appears before the URL parameter containing the QName which needs the prefix definition.

  As `<samlp:StatusCode>` elements may be nested hierarchically (see [[SAMLCore11]), there may exist multiple values for `<samlp:StatusCode>` in the response messages. These multiple values MUST be encoded by producing a URL-encoded space-separated string as the value of this query parameter. An example is as follows:

```
Value=samlp%3AResponder%20lib%3AFederationDoesNotExist
```

- Certain XML protocol messages support extensibility via an `<Extension>` element. Messages that are to be URL-encoded MUST adhere to the following restrictions when including extension content:
  - Only attribute values and elements with simple content models are permitted.
  - All attributes and elements MUST have an empty namespace and MUST have unique local names.
  - Each value included SHOULD NOT exceed 80 bytes in length (including encoding overhead).

XML digital signatures are not directly URL-encoded due to space concerns. If the Liberty XML protocol message is signed with an XML signature, the encoded URL form of the message MUST be signed as follows:

- Include the signature algorithm identifier as a new `<query>` component parameter named SigAlg, but omit the signature.
- Sign the string containing the URL-encoded message. The string to be signed MUST include only the `<query>` part of the URL (that is, everything after `?` and before `&Signature=`). Any required URL-escaping MUST be done before signing.
- Encode the signature using base64 (see [RFC2045]).
- Add the base64-encoded signature to the encoded message as a new data item named `Signature`.

Note that some characters in the base64-encoded signature value may require URL escaping before insertion into the URL `<query>` part, as is the case for any other data item value.

Any items added after the `Signature` `<query>` component parameter are implicitly unsigned.

The service URL provided by the provider (the URL to which `<query>` parameters are added) MUST NOT contain any pre-existing `<query>` parameter values.

The following signature algorithms (i.e., `DSAwithSHA1`, `RSAwithSHA1`) and their identifiers (the URIs) MUST be supported:

- `DSAwithSHA1` - `http://www.w3.org/2000/09/xmldsig#dsa-sha1`
- `RSAwithSHA1` - `http://www.w3.org/2000/09/xmldsig#rsa-sha1`

3.1.2.1.1. Size Limitations

When the request initiator knows or suspects that the user agent cannot process the full URL-encoded message in the URL due to size considerations, the requestor MAY send the Liberty XML protocol message using a form POST. The form MUST be constructed with contents that contain the field `LAREQ` or `LARES` with the respective value being the Liberty XML protocol request or response message (e.g., `<lib:AuthnRequest>` or `<lib:AuthnResponse>`) as defined in [LibertyProtSchema]. The Liberty XML protocol message MUST be encoded by applying a base64 transformation (refer to [RFC2045]) to the XML message and all its elements.

3.1.2.1.2. URL-encoded <lib:AuthnRequest>

The original `<lib:AuthnRequest>` message:

```
<lib:AuthnRequest RequestID="[RequestID]"
   MajorVersion="[MajorVersion]"
   MinorVersion="[MinorVersion]"
   IssueInstant="[IssueInstant]"
   consent="[consent]">
   <lib:ProviderID>[ProviderID]</lib:ProviderID>
   <lib:AffiliationID>[AffiliationID]</lib:AffiliationID>
   <lib:ForceAuthn>[ForceAuthn]</lib:ForceAuthn>
   <lib:IsPassive>[IsPassive]</lib:IsPassive>
   <lib:NameIDPolicy>[NameIDPolicy]</lib:NameIDPolicy>
   <lib:ProtocolProfile>[ProtocolProfile]</lib:ProtocolProfile>
   <lib:AssertionConsumerServiceID>[AssertionConsumerServiceID]
   </lib:AssertionConsumerServiceID>
   <lib:AuthnContext>
      <lib:AuthnContextStatementRef>[AuthnContextStatementRef]
      </lib:AuthnContextStatementRef>
   </lib:AuthnContext>
   <lib:RelayState>[RelayState]</lib:RelayState>
   <lib:AuthnContextComparison>[AuthnContextComparison]</lib:AuthnContextComparison>
```

```
            <lib:Scoping>
               <lib:ProxyCount>[ProxyCount]</lib:ProxyCount>
               <lib:IDPList>
                  <lib:IDPEntries>[IDPEntries]</lib:IDPEntries>
                  <lib:GetComplete>[GetComplete]</lib:GetComplete>
               </lib:IDPList>
            </lib:Scoping>
         </lib:AuthnRequest>
```

- Data elements that MUST be included in the encoded data with their values as indicated in brackets above if present in the original message:

  RequestID, MajorVersion, MinorVersion, IssueInstant, ProviderID, AffiliationID, ForceAuthn, IsPassive, NameIDPolicy, ProtocolProfile, AuthnContextStatementRef, AuthnContextClassRef, AuthnContextComparison, RelayState, ProxyCount, IDPEntries, GetComplete, consent.

- The <IDPEntries> element may contain multiple <IDPEntry> elements, each of which may contain multiple pieces of data (<ProviderID>, <ProviderName> and <Loc>). The <IDPEntries> element MUST be URL-encoded by taking only the <ProviderID> element from each individual <IDPEntry> element, and concatenating them in a space-separated string, as in the following example:

  ... &IDPEntries=http%3A%2F%2Fidp1.com%2Fliberty%2F%20http%3A%2F%2Fidp2.com%2Fliberty%2F ...

  The recipient of such a URL-encoded list of <ProviderID> elements may obtain the remainder of the information present in the original <IDPEntry> by accessing metadata for the individual providers referenced in the URL-encoded list.

- Example of <lib:AuthnRequest> message URL-encoded and signed:

```
http://idp.example.com/authn?RequestID=RMvY34pg%2FV9aGJ5yw0HL0AcjcqQF
&MajorVersion=1&MinorVersion=2&IssueInstant=2002-05-15T00%3A58%3A19
&consent=urn%3Aliberty%3Aconsent%3Aobtained&ProviderID=http%3A%2F%2Fsp.example.com%2Fliberty%2F

&ForceAuthn=true&IsPassive=false&NameIDPolicy=federated
&ProtocolProfile=http%3A%2F%2Fprojectliberty.org%2Fprofiles%2Fbrws-post
http%3A%2F%2Fwww.projectliberty.org%2Fschemas%2Fauthctx%2Fclasses%2FPasswordProtectedTransport
&RelayState=03mhakSms5tMQ0WRDCEzpF7BNcywZa75FwIcSSEPvbkoFxaQHCuNnc5yChId
DlWc7JBV9Xbw3avRBK7VFsPl2X
&SigAlg=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23rsa-sha1
&Signature=EoD8bNr2jEOe%2Fumon6oU%2FZGIIF7gbJAe4MLUUMrD%2B%3F7f8Yf3gfdZG2qPJdNAJkzVHGfO8W8DzpQ
%0D%0AsETTd5VP9MLPcvxbFQoF0CJJmvL26cPsuc54q7ojrcFCjJ%2F2CkDc4DAlYlZ5kPIq%2BtrykqLz0U%2BS%0D%
0ANqpNHkjh6W3YkGv7RBs%3D
```

3.1.2.1.3. URL-Encoded <lib:FederationTerminationNotification>

The original <lib:FederationTerminationNotification> message:

```
<lib:FederationTerminationNotification ...
   RequestID="[RequestID]"
   MajorVersion="[MajorVersion]"
   MinorVersion="[MinorVersion]"
   IssueInstant="[IssueInstant]"
   consent="[consent]">
   <lib:ProviderID>[ProviderID]</lib:ProviderID>
   <saml:NameIdentifier
      NameQualifier="[NameQualifier]"
 Format="[NameFormat]">[NameIdentifier]</saml:NameIdentifier>
</lib:FederationTerminationNotification>
```

- Data elements that MUST be included in the encoded data with their values as indicated in brackets above if present in the original message:

  RequestID, MajorVersion, MinorVersion, IssueInstant, ProviderID, NameQualifier, NameFormat, NameIdentifier, consent.

3.1.2.1.4. URL-Encoded <lib:LogoutRequest>

The original <lib:LogoutRequest> message:

```
<lib:LogoutRequest ...
    RequestID="[RequestID]"
    MajorVersion="[MajorVersion]"
    MinorVersion="[MinorVersion]"
    IssueInstant="[IssueInstant]"
    consent="[consent]">
      <lib:ProviderID>[ProviderID]</lib:ProviderID>
      <saml:NameIdentifier
         NameQualifier="[NameQualifier]"
         Format="[NameFormat]">
            [NameIdentifier]
      </saml:NameIdentifier>
      <lib:SessionIndex>[SessionIndex]</lib:SessionIndex>
      <lib:RelayState>[RelayState]</lib:RelayState>
   </lib:LogoutRequest>
```

- Data elements that MUST be included in the encoded data with their values as indicated in brackets above if present in the original message:

  RequestID, MajorVersion, MinorVersion, IssueInstant, ProviderID, NameQualifier, NameFormat, NameIdentifier, SessionIndex, RelayState, consent.

3.1.2.1.5. URL-Encoded <lib:LogoutResponse>

The <lib:LogoutResponse> response message:

```
<lib:LogoutResponse
   ResponseID="[ResponseID]"
   InResponseTo="[InResponseTo]"
   MajorVersion="[MajorVersion]"
   MinorVersion="[MinorVersion]"
   IssueInstant="[IssueInstant]"
   Recipient="[Recipient]">
 <lib:ProviderID>[ProviderID]</lib:ProviderID>
 <samlp:Status>
 <samlp:StatusCode Value="[Value]"/>
 </samlp:Status>
 <lib:RelayState>[RelayState]</lib:RelayState>
</lib:LogoutResponse>
```

- Data elements that MUST be included in the encoded data with their values as indicated in brackets above if present in the original message:

  ResponseID, InResponseTo, MajorVersion, MinorVersion, IssueInstant, Recipient, ProviderID, Value, RelayState.

- The <lib:LogoutResponse> message may contain nested status code information. Multiple values MUST be URL-encoded by creating a space-separated list (see general requirements at top of Section 3.1.2.1.5).

3.1.2.1.6. URL-Encoded <lib:RegisterNameIdentifierRequest>

The original <lib:RegisterNameIdentifierRequest> message:

```
<lib:RegisterNameIdentifierRequest
  RequestID="[RequestID]"
  MajorVersion="[MajorVersion]"
  MinorVersion="[MinorVersion]"
  IssueInstant="[IssueInstant]">
  <lib:ProviderID>[ProviderID]</lib:ProviderID>
  <lib:IDPProvidedNameIdentifier
    NameQualifier="[IDPNameQualifier]"
    Format="[IDPNameFormat]">[IDPProvidedNameIdentifier]
  </lib:IDPProvidedNameIdentifier>
  <lib:SPProvidedNameIdentifier
    NameQualifier="[SPNameQualifier]"
    Format="[SPNameFormat]">[SPProvidedNameNameIdentifier
  </lib:SPProvidedNameIdentifier>
  <lib:OldProvidedNameIdentifier
    NameQualifier="[OldNameQualifier]"
    Format="[OldNameFormat]">[OldProvidedNameIdentifier]
  </lib:OldProvidedNameIdentifier>
  <lib:RelayState>[RelayState]</lib:RelayState>
</lib:RegisterNameIdentifierRequest>
```

- Data elements that MUST be included in the encoded data with their values as indicated in brackets above if present in the original message:

```
RequestID, MajorVersion, MinorVersion, IssueInstant,
ProviderID, IDPNameQualifier, IDPNameFormat, IDPProvidedNameIdentifier,
SPNameQualifier, SPNameFormat, SPProvidedNameIdentifier,
OldNameQualifier, OldNameFormat, OldProvidedNameIdentifier,
RelayState
```

3.1.2.1.7. URL-Encoded <lib:RegisterNameIdentifierResponse>

The <lib:RegisterNameIdentifierResponse> response message:

```
<lib:RegisterNameIdentifierResponse
  ResponseID="[ResponseID]"
  InResponseTo="[InResponseTo]"
  MajorVersion="[MajorVersion]"
  MinorVersion="[MinorVersion]"
  IssueInstant="[IssueInstant]"
  Recipient="[Recipient]">
  <lib:ProviderID>[ProviderID]</lib:ProviderID>
  <samlp:Status>
  <samlp:StatusCode Value="[Value]"/>
  </samlp:Status>
  <lib:RelayState>[RelayState]</lib:RelayState>
</lib:RegisterNameIdentifierResponse>
```

- Data elements that MUST be included in the encoded data with their values as indicated in brackets above if present in the original message:

  `ResponseID, InResponseTo, MajorVersion, MinorVersion, IssueInstant, Recipient, ProviderID, Value, RelayState`

- The `<lib:RegisterNameIdentifierResponse>` message may contain nested status code information. Multiple values MUST be URL-encoded by creating a space-separated list (see general requirements at top of Section 3.1.2.1.

3.1.3. Provider Metadata

The majority of the Liberty profiles defined in this document rely on metadata that specify the policies that govern the behavior of the service provider or identity provider. These provider metadata may be shared out of band between an identity provider and a service provider prior to the exchange of Liberty protocol messages or with the protocols described in [LibertyMetadata]. The provider metadata relevant to each profile are listed in this document at the beginning of the profile category. Refer to [LibertyMetadata] for a complete enumeration of the Liberty provider metadata elements and their associated schema.

3.2. Single Sign-On and Federation Profiles

This section defines the profiles by which a service provider obtains an authentication assertion of a user agent from an identity provider to facilitate single sign-on. Additionally, the single sign-on profiles can be used as a means of federating an identity from a service provider to an identity provider through the use of the `<NameIDPolicy>` element in the `<lib:AuthnRequest>` protocol message as specified in [LibertyProtSchema].

The single sign-on profiles make use of the following metadata elements, as defined in [LibertyProtSchema]:

- `ProviderID` Used to uniquely identify the service provider to the identity provider and is documented in these profiles as "service provider ID."
- `AffiliationID` Used to uniquely identify an affiliation group to the identity provider and is documented in these profiles as "affiliation ID."
- `SingleSignOnServiceURL` The URL at the identity provider that the service provider should use when sending single sign-on and federation requests. It is documented in these profiles as "single sign-on service URL."
- `AssertionConsumerServiceURL` The URL(s) at the service provider that an identity provider should use when sending single sign-on or federation responses. It is documented in these profiles as "assertion consumer service URL."
- `SOAPEndpoint` The SOAP endpoint location at the service provider or identity provider to which Liberty SOAP messages are sent.

3.2.1. Common Interactions and Processing Rules

This section defines the set of interactions and process rules that are common to all single sign-on profiles.

All single sign-on profiles can be described by one interaction diagram, provided that different messages are optional in different profiles and that the actual content of the messages may differ slightly. Where interactions and messages differ or are optional, they are designated and detailed within the specific single sign-on profiles. Figure 1 represents the basic template of interactions for achieving single sign-on. This should be used as the baseline for all single sign-on profiles.

In the figure below, steps 1 through 5 can be considered typical but optional. An identity provider MAY initiate a SSO profile by unilaterally creating a `<lib:AuthnResponse>` or artifact, and proceeding with step 6, as discussed in [LibertyProtSchema].

It should be noted that multiple identity providers may be involved in the authentication of the Principal. Although a single identity provider is depicted in the profiles below (Figure 1, that identity provider MAY interact with other identity providers to authenticate the Principal using the proxying method described in [LibertyProtSchema] and the profiles as noted below. In such situations these profiles would be used by the identity provider originally contacted by the requesting service provider to communicate with additional identity providers.

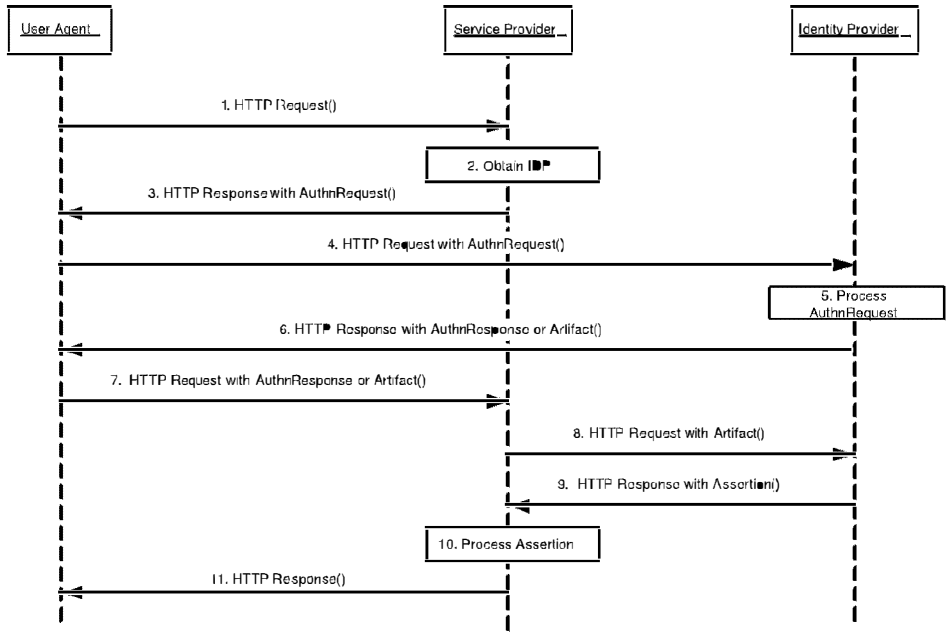


Figure 1. Basic single sign-on profile.

3.2.1.1. Step 1: HTTP Request

In step 1, the user agent accesses the intersite transfer service at the service provider with information about the desired target attached to the URL. Typically, access to the intersite transfer service occurs via a redirection by the service provider in response to a user agent request for a restricted resource.

It is RECOMMENDED that the HTTP request be made over either SSL 3.0 (see [SSL]) or TLS 1.0 (see [RFC2246]) to maintain confidentiality and message integrity in step 1.

3.2.1.2. Step 2: Obtain Identity Provider

In step 2, the service provider obtains the address of the appropriate identity provider to redirect the user agent to in step 3. The means by which the identity provider address is obtained is implementation-dependent and up to the service provider. The service provider MAY use the Liberty identity provider introduction profile in this step.

3.2.1.3. Step 3: HTTP Response with <AuthnRequest>

In step 3, the service provider's intersite transfer service responds and sends the user agent to the single sign-on service URL at the identity provider. The form and contents of the HTTP response in this step are profile-dependent.

3.2.1.4. Step 4: HTTP Request with <AuthnRequest>

In step 4, the user agent accesses the identity provider's single sign-on service URL with the `<lib:AuthnRequest>` information. This request may be a GET or POST request; providers MUST support both methods. As described later, such a POST MUST contain an `LAREQ` form element containing the XML protocol request in base64-encoded format.

3.2.1.5. Step 5: Processing <AuthnRequest>

In step 5, the identity provider MUST process the `<lib:AuthnRequest>` message according to the rules specified in [LibertyProtSchema].

If the Principal has not yet been authenticated with the identity provider, authentication at the identity provider MAY occur in this step. The identity provider MAY obtain consent from the Principal for federation, or otherwise consult the Principal. To this end the identify provider MAY return to the HTTP request any HTTP response, including but not limited to HTTP Authentication, HTTP redirect, or content. The identity provider SHOULD respect the HTTP User-Agent and Accept headers and SHOULD avoid responding with content-types that the User-Agent may not be able to accept. Authentication of the Principal by the identity provider is dependent upon the `<lib:AuthnRequest>` message content.

In case the identity provider responds to the user agent with a form, it is RECOMMENDED that the `<input>` parameters of the form be named according to [RFC3106] whenever possible.

3.2.1.6. Step 6: HTTP Response with <AuthnResponse> or Artifact

In step 6, the identity provider MUST respond to the user agent with a `<lib:AuthnResponse>`, a SAML artifact, or an error. The form and contents of the HTTP response in this step are profile-dependent.

3.2.1.7. Step 7: HTTP Request with <AuthnResponse> or Artifact

In step 7, the user agent accesses the assertion consumer service URL at the service provider with a `<lib:AuthnResponse>` or a SAML artifact. This request may be a GET or POST request; providers MUST support both methods. As described later, such a POST MUST contain an `LARES` form element containing the XML protocol request or artifact in base64-encoded format.

3.2.1.8. Step 8: HTTP Request with Artifact

Step 8 is required only for single sign-on profiles that use a SAML artifact.

In this step the service provider, in effect, dereferences the single SAML artifact in its possession to acquire the authentication assertion that corresponds to the artifact.

The service provider MUST send a `<samlp:Request>` SOAP message to the identity provider's SOAP endpoint, requesting the assertion by supplying the SAML assertion artifact in the `<samlp:AssertionArtifact>` element as specified in [SAMLBind11].

The service provider MUST provide a mechanism for the identity provider to authenticate the service provider.

3.2.1.9. Step 9: HTTP Response with Assertion

Step 9 is required only for single sign-on profiles that use a SAML artifact.

In this step if the identity provider is able to find or construct the requested assertion, it responds with a `<samlp:Response>` SOAP message with the requested `<saml:Assertion>`. Otherwise, it returns an appropriate status code, as defined within the "SOAP binding for SAML" (see[SAMLBind11]) and the [LibertyProtSchema].

3.2.1.10. Step 10: Process Assertion

In step 10, the service provider processes the `<saml:Assertion>` returned in the `<samlp:Response>` or `<lib:AuthnResponse>` protocol message to determine its validity and how to respond to the Principal's original request. The signature on the `<saml:Assertion>` must be verified.

The service provider processing of the assertion MUST adhere to the rules defined in [SAMLCore11] for things such as assertion `<saml:Conditions>` and `<saml:Advice>`.

The service provider MAY obtain authentication context information for the Principal's current session from the `<lib:AuthnContext>` element contained in `<saml:Advice>`. Similarly, the information in the `<lib:RelayState>` element MAY be obtained and used in further processing by the service provider.

3.2.1.11. Step 11: HTTP Response

In step 11, the user agent is sent an HTTP response that either allows or denies access to the originally requested resource.

3.2.2. Liberty Artifact Profile

The Liberty artifact profile relies on a reference to the needed assertion traveling in a SAML artifact, which the service provider must dereference from the identity provider to determine whether the Principal is authenticated. This profile is an adaptation of the "Browser/artifact profile" for SAML as documented in [SAMLBind11]. See Figure 3.

The following URI-based identifier MUST be used when referencing this specific profile (for example, `<lib:ProtocolProfile>` element of the `<lib:AuthnRequest>` message):

URI: `http://projectliberty.org/profiles/brws-art`

The Liberty artifact profile consists of a single interaction among three parties: a user agent, an identity provider, and a service provider, with a nested subinteraction between the identity provider and the service provider.

3.2.2.1. Interactions

Figure 2 illustrates the Liberty artifact profile for single sign-on.

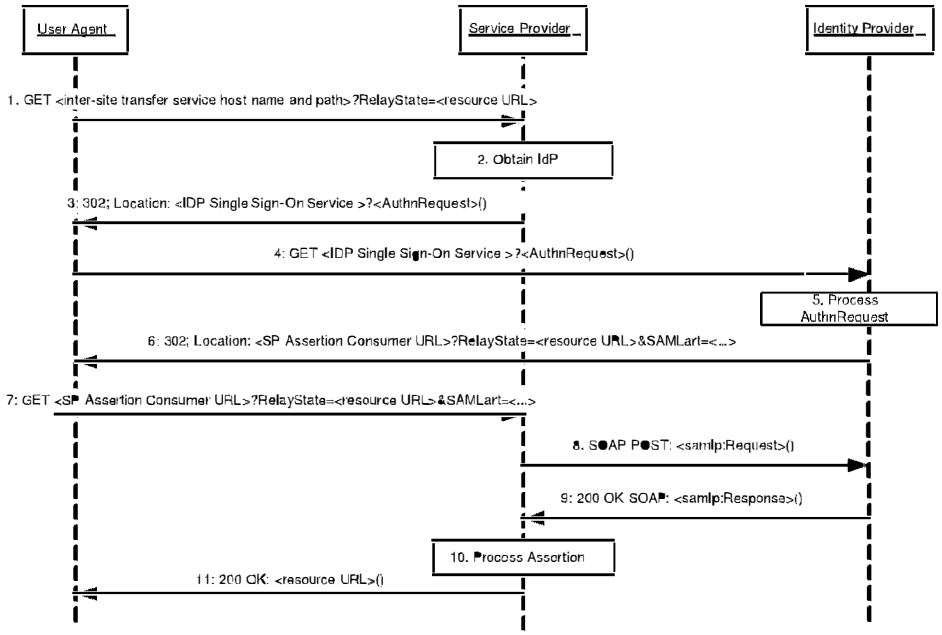


Figure 2. Liberty artifact profile for single sign-on

This profile description assumes that the user agent has already authenticated at the identity provider prior to step 1. Thus, a valid session exists for the user agent at the identity provider. When implementing this profile, all processing rules defined in Section 3.2.1 for the single sign-on profiles MUST be followed. Additionally, the following rules MUST be observed as they relate to steps 3, 6 and 7:

3.2.2.1.1. Step 3: Single sign on Service with <AuthnRequest>

In step 3, the service provider's intersite transfer service responds and instructs the user agent to access the single sign-on service URL at the identity provider.

This step may take place via an HTTP 302 redirect, a WML redirect deck or any other method that results in the user agent being instructed to make an HTTP GET or POST request to the identity provider's single signon service.

This response MUST adhere to the following rules:

- The response MUST contain the identity provider's single sign-on service URL (for example, as the Location header of an HTTP 302 redirect, the `action` attribute of an HTML `form` or the `href` attribute of a `<go>` element in a WML redirect deck).

- The identity provider's single sign-on service URL MUST specify `https` as the URL scheme.

  Note:

  Future protocols may be adopted and enabled to work within this framework. Therefore, implementers are encouraged to not hardcode a reliance on `https`.

- The response MUST include one of the following:

- A <query> component containing the <lib:AuthnRequest> protocol message as defined in [LibertyProtSchema] with formatting as specified in Section 3.1.2.

  Note:

  The <lib:RelayState> element of the <lib:AuthnRequest> message can be used by the service provider to help maintain state information during the single sign-on and federation process. For example, the originally requested resource (i.e., RelayState in step 1) could be stored as the value for the <lib:RelayState> element, which would then be returned to the service provider in the <lib:AuthnResponse> in step 7. The service provider could then use this information to formulate the HTTP response to the user agent in step 11.

- An HTTP form containing the field LAREQ with the value of the <lib:AuthnRequest> protocol message as defined in [LibertyProtSchema]. The <lib:AuthnRequest> MUST be encoded by applying a base64 transformation (see [RFC2045].

Implementation examples:

- HTTP 302 Redirect

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location: https://<Identity Provider Single Sign-On Service host name and path>?<query>
<other HTTP 1.0 or 1.1 components>
```

- HTML Form POST

```
<html>
   <body onLoad="document.forms[0].submit()">
      <form action="https://<Identity Provider Single Sign-On Service host name and path>"
method="POST">
         <input type="hidden" name="LAREQ" value="<base64 encoded AuthnRequest>" >
      </form>
   </body>
</html>
```

- WML Redirect with POST

```
                ...
<wml>
<card id="redirect" title="Log In">
   <onenterforward>
      <go method="post" href="<Identity Provider Single Sign-On service host name and path>" >
         <postfield name="LAREQ" Value="<base64-encoded AuthnRequest>" />
      </go>
   </onenterforward>
   <onenterbackward>
      <prev/>
   </onenterbackward>
   <p>
      Contacting IdP. Please wait...
   </p>
...
</card>
...
</wml>
```

- WML Redirect with GET

```
                ...
<wml>
<card id="redirect" title="Log In">
   <onenterforward>
      <go href="<Identity Provider Single Sign-On service host name and path>?<query>" />
   </onenterforward>
   <onenterbackward>
      <prev/>
   </onenterbackward>
   <p>
      Contacting IdP. Please wait...
   </p>
...
</card>
...
</wml>
```

where:

`<Identity Provider Single Sign-On service host name and path>`

This element provides the host name, port number, and path components of the single sign-on service URL at the identity provider.

`<query>= ...<URL-encoded AuthnRequest> ...`

A `<query>` component MUST contain a single authentication request:

`<base64-encoded AuthnRequest>`

A `<base64-encoded AuthnRequest>` component MUST contain a single authentication request message in base64-encoded form.

3.2.2.1.2. Step 6: Redirecting to the Service Provider

In step 6, the identity provider instructs the user agent to access the service provider's assertion consumer service URL, and provides a SAML artifact for de-referencing by the service provider.

This step may take place via an HTTP 302 redirect, a WML redirect deck or any other method that results in the user agent being instructed to make an HTTP GET or POST request to the service provider's assertion consumer service.

This response MUST adhere to the following rules:

- The response MUST contain the service provider's assertion consumer service URL (for example, as the Location header of an HTTP 302 redirect, the `action` attribute of an HTML`form` or the `href` attribute of a `<go>` element in a WML redirect deck).

- The service provider's assertion consumer service URL MUST specify `https` as the URL scheme.
  Note:
  Future protocols may be adopted and enabled to work within this framework. Therefore, implementers are encouraged to not hardcode a reliance on `https`.

- The response MUST include one of the following:

- A <query> component containing a parameter SAMLart, the value of which is the SAML artifact on success or on failure. In the case of failure, the status will be conveyed in the <saml:Response> returned in Step 9. Additionally, if the <lib:AuthnRequest> processed in Step 5 included a value for the <lib:RelayState> element, then a parameter named RelayState with a value set to that of the <lib:RelayState> element MUST be included in the <query> component.

- An HTTP form containing the field LARES with the value of the SAML Artifact as defined in Section 3.2.2.2 If a value for <RelayState> was supplied in the <lib:AuthnRequest>, then the form MUST contain a field RelayState, with a value obtained from that element in the <lib:AuthnRequest>.

- All SAML artifacts returned MUST contain the same identity provider ID.

Implementation examples:

- HTTP 302 Redirect

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location: https://<Service Provider Assertion Consumer Service host name and path>?<query>
<other HTTP 1.0 or 1.1 components>
```

- HTML Form POST

```
<html>
   <body onLoad="document.forms[0].submit()">
      <form action="https://<Service Provider Assertion Consumer Service host name and path>"
method="POST">
         <input type="hidden" name="LAREQ" value="<SAML Artifact>" >
         <input type="hidden" name="RelayState" value="<RelayState>" >
      </form>
   </body>
</html>
```

- WML Redirect with POST

```
                ...
<wml>
<card id="redirect" title="Artifact">
   <onenterforward>
      <go method="post" href="<Service Provider Assertion Consumer Service host name and path>" >
         <postfield name="LARES" Value="<SAML Artifact>" />
         <postfield name="RelayState" Value="<RelayState>" />
      </go>
   </onenterforward>
   <onenterbackward>
      <prev/>
   </onenterbackward>
   <p>
      Contacting IdP. Please wait...
   </p>
...
</card>
...
</wml>
```

- WML Redirect with GET

```
                ...
<wml>
<card id="redirect" title="Artifact">
   <onenterforward>
      <go href="<Service Provider Assertion Consumer Service host name and path>?<query>" />
   </onenterforward>
   <onenterbackward>
      <prev/>
   </onenterbackward>
   <p>
      Contacting IdP. Please wait...
   </p>
...
</card>
...
</wml>
```

where:

<Service Provider Assertion Consumer Service host name and path>

This element provides the host name, port number, and path components of the assertion consumer service URL at the service provider.

<query>= ...SAMLArt=<SAML Artifact> ...RelayState=<resource URL>

A <query> component MUST contain at least one SAML Artifact. A single RelayState MUST be included if a value for the <RelayState> was provided in the <lib:AuthnRequest>. All SAML Artifacts included MUST contain the same identity provider ID (see Section 3.2.2.2).

<SAML Artifact>

A <SAML Artifact> component MUST contain at least one SAML Artifact.

<RelayState>

A form field named RelayState, with the value of that element from the <lib:AuthnRequest> MUST be included if a value for the <RelayState> was provided in the <lib:AuthnRequest> and the HTTP request is made using a POST.

3.2.2.1.3. Step 7: Accessing the Assertion Consumer Service

In step 7, the user agent accesses the assertion consumer service URL at the service provider, with a SAML artifact representing the Principal's authentication information attached to the URL.

3.2.2.2. Artifact Format

The artifact format includes a mandatory two-byte artifact type code, as follows:

```
SAML_artifact    := B64(TypeCode RemainingArtifact)
TypeCode         := Byte1Byte2
```

The notation B64(TypeCode RemainingArtifact) represents the application of the base64 transformation to the catenation of the TypeCode and RemainingArtifact. This profile defines an artifact type of type code 0x0003,

which is REQUIRED (mandatory to implement) for any implementation of the Liberty browser artifact profile. This artifact type is defined as follows:

```
TypeCode        := 0x0003
RemainingArtifact := IdentityProviderSuccinctID AssertionHandle
IdentityProviderSuccinctID:= 20-byte_sequence
AssertionHandle   := 20-byte_sequence
```

`IdentityProviderSuccinctID` is a 20-byte sequence used by the service provider to determine identity provider identity and location. It is assumed that the service provider will maintain a table of `IdentityProviderSuccinctID` values as well as the URL (or address) for the corresponding SAML responder at the identity provider. This information is communicated between the identity provider and service provider out of band. On receiving the SAML artifact, the service provider determines whether the `IdentityProviderSuccinctID` belongs to a known identity provider and, if so, obtains the location before sending a SAML request.

Any two identity providers with a common service provider MUST use distinct `IdentityProviderSuccinctID` values. Construction of `AssertionHandle` values is governed by the principles that the values SHOULD have no predictable relationship to the contents of the referenced assertion at the identity provider, and that constructing or guessing the value of a valid, outstanding assertion handle MUST be infeasible.

The following rules MUST be followed for the creation of SAML artifacts at identity providers:

- Each identity provider selects a single identification URL, corresponding to the provider metadata element ProviderID specified in [LibertyMetadata].
- The identity provider constructs the `IdentityProviderSuccinctID` component of the artifact by taking the SHA-1 hash of the identification URL as a 20-byte binary value. Note that the `IdentityProviderSuccinctID` value, used to construct the artifact, is not encoded in hexadecimal. The `AssertionHandle` value is constructed from a cryptographically strong random or pseudo-random number sequence (see [RFC1750]) generated by the identity provider. The sequence consists of a value of at least eight bytes. The value should be padded to a total length of 20 bytes.

3.2.3. Liberty Browser POST Profile

The Liberty browser POST profile allows authentication information to be supplied to an identity provider without the use of an artifact. Figure 3 diagrams the interactions between parties in the Liberty POST profile. This profile is an adaptation of the "Browser/post profile" for SAML as documented in [SAMLBind11].

The following URI-based identifier MUST be used when referencing this specific profile (for example, `<lib:ProtocolProfile>` element of the `<lib:AuthnRequest>` message):

URI: `http://projectliberty.org/profiles/brws-post`

The Liberty POST profile consists of a series of two interactions, the first between a user agent and an identity provider, and the second directly between the user agent and the service provider.

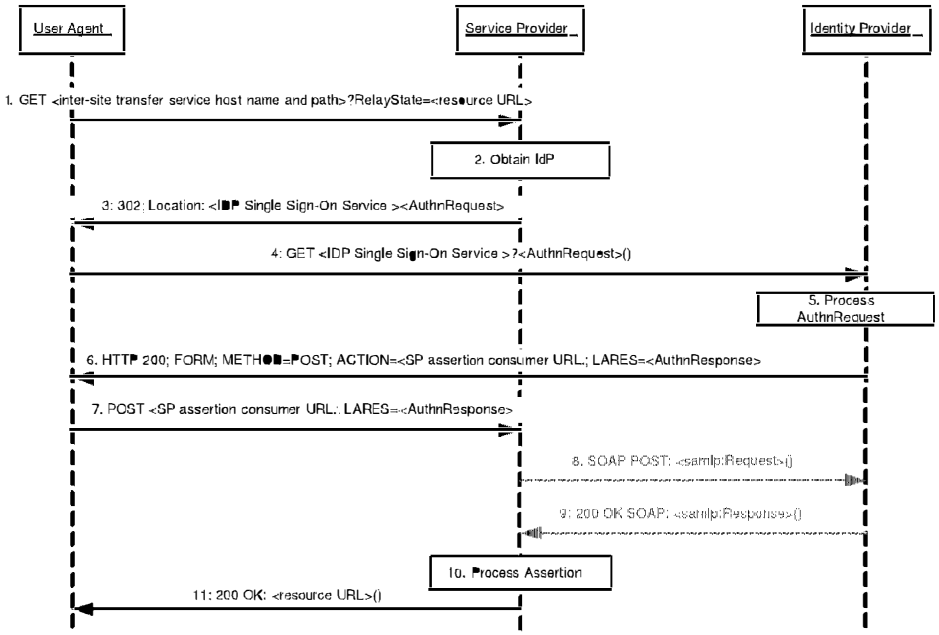


Figure 3. Liberty browser POST profile for single sign-on

This profile description assumes that the user agent has already authenticated at the identity provider prior to step 1. Thus, a valid session exists for the user agent at the identity provider.

When implementing this profile, all processing rules defined in Section 3.2.1 for single sign-on profiles MUST be followed with the exception that steps 8 and 9 MUST be omitted. Additionally, the following rules MUST be observed as they relate to steps 3, 6 and 7:

3.2.3.1. Step 3: Single Sign-On Service with <AuthnRequest>

In step 3, the service provider's intersite transfer service responds and sends the user agent to the single sign-on service URL at the identity provider.

This step may take place via an HTTP 302 redirect, a WML redirect deck or any other method that results in the user agent being instructed to make an HTTP GET or POST request to the identity provider's single sign-on service.

This response MUST adhere to the following rules:

- The response MUST contain the identity provider's single sign-on service URL (for example, as the Location header of an HTTP 302 redirect, the `action` attribute of an HTML `form` or the `href` attribute of a `<go>` element in a WML redirect deck).

- The identity provider's single sign-on service URL MUST specify `https` as the URL scheme.
  Note:
  Future protocols may be adopted and enabled to work within this framework. Therefore, implementers are encouraged to not hardcode a reliance on `https`.

- The response MUST include one of the following:

- A <query> component containing the <lib:AuthnRequest> protocol message as defined in [LibertyProtSchema] with formatting as specified in Section 3.1.2

  Note:

  The <lib:RelayState> element of the <lib:AuthnRequest> message can be used by the service provider to help maintain state information during the single sign-on and federation process. For example, the originally requested resource (that is, RelayState in step 1) could be stored as the value for the <lib:RelayState> element, which would then be returned to the service provider in the <lib:AuthnResponse> in step 7. The service provider could then use this information to formulate the HTTP response to the user agent in step 11.

- An HTTP form containing the field LARES with the value of the <lib:AuthnRequest> protocol message as defined in [LibertyProtSchema]. The <lib:AuthnRequest> MUST be encoded by applying a base64 transformation (see [RFC2045]).

See the discussion of this step in the artifact profile for implementation examples.

3.2.3.2. Step 6: Generating and Supplying the <AuthnResponse>

In step 6 the identity provider generates an HTML form containing an authentication assertion that MUST be sent in an HTTP 200 response to the user agent.

The form MUST be constructed such that it requests a POST to the service provider's assertion consumer URL with form contents that contain the field LARES with the value being the <lib:AuthnResponse> protocol message as defined in [LibertyProtSchema]. The <lib:AuthnResponse> MUST be encoded by applying a base64 transformation (refer to [RFC2045]) to the <lib:AuthnResponse> and all of its elements. The service provider's assertion consumer service URL used as the target of the form POST MUST specify https as the URL scheme; if another scheme is specified, it MUST be treated as an error by the identity provider.

Multiple <saml:Assertion> elements MAY be included in the response. The identity provider MUST digitally sign each of the assertions included in the response.

The <saml:ConfirmationMethod> element of the assertion MUST be set to the value specified in [SAMLCore11] for "Assertion Bearer."

3.2.3.3. Step 7: Posting the Form Containing the <AuthnResponse>

In step 7 the user agent issues the HTTP POST request containing the <lib:AuthnResponse> to the service provider.

3.2.4. Liberty-Enabled Client and Proxy Profile

The Liberty-enabled client and proxy profile specifies interactions between Liberty-enabled clients and/or proxies, service providers, and identity providers. See Figure 5. A Liberty-enabled client is a client that has, or knows how to obtain, knowledge about the identity provider that the Principal wishes to use with the service provider. In addition a Liberty-enabled client receives and sends Liberty messages in the body of HTTP requests and responses. Therefore, Liberty-enabled clients have no restrictions on the size of the Liberty protocol messages.

A Liberty-enabled proxy is an HTTP proxy (typically a WAP gateway) that emulates a Liberty-enabled client. Unless stated otherwise, all statements referring to "LECP" are to be understood as statements about both Liberty-enabled clients and Liberty-enabled proxies.

In some environments the successful deployment of a Liberty-Enabled proxy may require that service providers in those environments perform operations in addition to those described below. Such cases, and specific guidance for them, are covered in [LibertyImplGuide].

The following URI-based identifier must be used when referencing this specific profile (for example, <lib:ProtocolProfile> element of the <lib:AuthnRequest> message):

URI: http://projectliberty.org/profiles/lecp

A LECP, in addition to meeting the common requirements for profiles in Section 3.1, MUST indicate that it is a LECP by including a Liberty-Enabled header or entry in the value of the HTTP User-Agent header for each HTTP request it makes. The preferred method is the Liberty-Enabled header. The formats of the Liberty-Enabled header and User-Agent header entry are defined in Section 3.2.4.1.

3.2.4.1. Liberty-Enabled Indications

A LECP SHOULD add the Liberty-Enabled header to each HTTP request. The Liberty-Enabled header MUST be named `Liberty-Enabled` and be defined as using Augmented BNF as specified in section 2 of [RFC2616].

```
Liberty-Enabled = "Liberty-Enabled" ":" LIB_Version ["," 1#Extension]
LIB_Version = "LIB-V" "=" 1*absoluteURI
; any spaces or commas in the absoluteURI MUST be escaped as defined in section 2.4 of [RFC 2396]
Extension = ExtName "=" ExtValue
ExtName = {["." host] | <any field-value but ".", "," or "="> } <any field-value but "=" or ",">
ExtValue = <any field-value but ",">)
```

The comment, field-value, and product productions are defined in [RFC2616]. `LIB_Version` identifies the versions of the Liberty specifications that are supported by this LECP. Each version is identified by a URI. Service providers or identity providers receiving a Liberty-Enabled header MUST ignore any URIs listed in the `LIB_Version` production that they do not recognize. All LECPs compliant with this specification MUST send out, at minimum, the URI `http://projectliberty.org/specs/v1` as a value in the `LIB_Version` production. It SHOULD precede this with the URI `urn:liberty:iff:2003-08` if it supports version 1.2 requests and knows that the identity providers available to it also support version 1.2 requests and responses. It MUST NOT include this URI if it knows that the identity providers available to it cannot process version 1.2 messages. The ordering of the URIs in the `LIB_Version` header is meaningful; therefore, service providers and identity providers are encouraged to use the first version in the list that they support. Supported Liberty versions are not negotiated between the LECP and the service provider. The LECP advertises what version it supports; the service provider MUST return the response for the corresponding version as defined in step 3 below.

Optional extensions MAY be added to the Liberty-Enabled header to indicate new information. The value of the `ExtName` production MUST use the `"host" ";"` prefixed form if the new extension name has not been standardized and registered with Liberty or its designated registration authorities. The value of the host production MUST be an IP or DNS address that is owned by the issuer of the new name. By using the DNS/IP prefix, effectively namespace collisions can be prevented without the need of introducing another centralized registration agency.

A LECP MAY include the Liberty-Agent header in its requests. This header provides information about the software implementing the LECP functionality and is similar to the User-Agent and Server headers in HTTP.

```
Liberty-Agent = "Liberty-Agent" ":" 1*( product | comment)
```

Note:

The reason for introducing the new header (that is, Liberty-Enabled) rather than using User-Agent is that a LECP may be a Liberty-enabled proxy. In such a case the information about the Liberty-enabled proxy would not be in the User-Agent header. In theory the information could be in the VIA header. However, for security reasons, values in the VIA header can be collapsed, and comments (where software information would be recorded) can always be removed. As such, the VIA header is not suitable. Using the User-Agent header for a Liberty-enabled client and the Liberty-Agent header for a Liberty-enabled proxy was also discussed. However, this approach seemed too complex.

Originally the Liberty-Agent header was going to be part of the Liberty-Enabled header. However, header lengths in HTTP implementations are limited; therefore, putting this information in its own header was considered the preferred approach.

A LECP MAY add a Liberty-Enabled entry in the HTTP User-Agent request header. The HTTP User-Agent header is specified in [RFC2616]. A LECP MAY include in the value of this header the `Liberty-Enabled` string as defined above for the Liberty-Enabled header.

Note:

The reason for adding information to the User-Agent header is to allow for Liberty-enabled client products that must rely on a platform that cannot be instructed to insert new headers in each HTTP request.

The User-Agent header is often overloaded; therefore, the Liberty-Enabled header should be the first choice for any implementation of a LECP. The entry in the User-Agent header then remains as a last resort.

3.2.4.2. Interactions

Figure 5 illustrates the Liberty-enabled client and proxy profile for single sign-on.

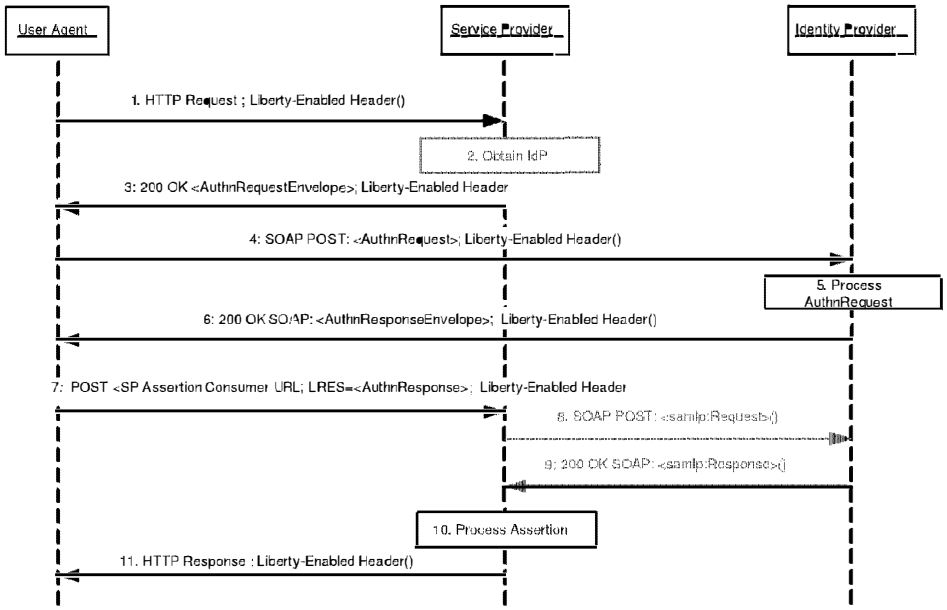


Figure 5. Liberty-enabled client and proxy profile for single sign-on

This profile description assumes that the user agent has already authenticated at the identity provider prior to step 1. Thus, a valid session exists for the user agent at the identity provider.

The LECP receives authentication requests from the service provider in the body of the HTTP response. The LECP submits this authentication request as a SOAP request to the identity provider. Because this SOAP request is between the LECP and the identity provider, TLS authentication cannot be performed between service provider and identity provider; therefore, service providers and identity providers MUST rely on the signature of the `<lib:AuthnRequest>` and the returned `<saml:Assertion>`, respectively, for mutual authentication.

When implementing this profile, processing rules for steps 5, 10, and 11 defined in Section 3.2.1 for single sign-on profiles MUST be followed, while steps 2, 8, and 9 MUST be omitted. Additionally, the following rules MUST be observed as they relate to steps 1, 3, 4, 6, and 7:

3.2.4.2.1. Step 1: Accessing the Service Provider

In step 1, the user agent accesses the service provider with the Liberty-Enabled header (or with the Liberty-Enabled entry in the User-Agent header) included in the HTTP request.

The HTTP request MUST contain only one Liberty-Enabled header. Hence if a proxy receives an HTTP request that contains a Liberty-Enabled header, it MUST NOT add another Liberty-Enabled header. However, a proxy MAY replace the Liberty-Enabled header. A proxy that replaces or adds a Liberty-Enabled header MUST process `<lib:AuthnRequest>` messages as defined in steps 3 and 4 as well as `<lib:AuthnResponse>` messages as specified in steps 6 and 7.

It is RECOMMENDED that a LECP add "`application/vnd.liberty-request+xml`" as one of its supported content types to the Accept header.

3.2.4.2.2. Step 3: HTTP Response with <AuthnRequest>

In step 3, the service provider's intersite transfer service issues an HTTP 200 OK response to the user agent. The response MUST contain a single `<lib:AuthnRequestEnvelope>` with content as defined in [LibertyProtSchema]. If a service provider receives a Liberty-Enabled header, or a User-Agent header with the Liberty-Enabled entry, the service provider MUST respond according to the Liberty-enabled client and proxy profile and include a Liberty-Enabled header in its response. Hence service providers MUST support the Liberty-enabled client and proxy profile.

The processing rules and default values for the Liberty-Enabled indications are as defined in Section 3.2.4.1. The service provider MAY advertise any Liberty version supported in this header, not only the version used for the specific response.

The HTTP response MUST contain a Content-Type header with the value `application/vnd.liberty-request+xml` unless the LECP and service provider have negotiated a different format.

A service provider MAY provide a list of identity providers it recognizes by including the `<lib:IDPList>` element in the `<lib:AuthnRequestEnvelope>`. The format and processing rules for the identity provider list MUST be as defined in [LibertyProtSchema].

Note:

In cases where a value for the `<lib:GetComplete>` element is provided within `<lib:IDPList>`, the URI value for this element MUST specify `https` as the URL `<scheme>`.

The service provider MUST specify a URL for receiving `<AuthnResponse>` elements, locally generated by the intermediary, by including the `<lib:AssertionConsumerServiceURL>` element in the `<lib:AuthnRequestEnvelope>`.

The following example demonstrates the usage of the `<lib:AuthnRequestEnvelope>`:

```
<?xml version="1.0" ?>
<lib:AuthnRequestEnvelope xmlns:lib="urn:liberty:iff:2003-08">
   <lib:AuthnRequest >
      . . . AuthnRequest goes here . . .
   </lib:AuthnRequest>
   <lib:AssertionConsumerServiceURL>
      https://service-provider.com/LibertyLogin
   </lib:AssertionConsumerServiceURL>
   <lib:IDPList >
      . . . IdP list goes here . . .
   </lib:IDPList>
</lib:AuthnRequestEnvelope>
```

If the service provider does not support the LECP-advertised Liberty version, the service provider MUST return to the LECP an HTTP 501 response with the reason phrase "Unsupported Liberty Version."

The responses in step 3 and step 6 SHOULD NOT be cached. To this end service providers and identity providers SHOULD place both "`Cache-Control: no-cache`" and "`Pragma: no-cache`" on their responses to ensure that the LECP and any intervening proxies will not cache the response.

3.2.4.2.3. Step 4: HTTP Request with <AuthnRequest>

In step 4, the LECP determines the appropriate identity provider to use and then issues an HTTP POST of the `<lib:AuthnRequest>` in the body of a SOAP message to the identity provider's single sign-on service URL. The request MUST contain the same `<lib:AuthnRequest>` as was received in the `<lib:AuthnRequestEnvelope>` from the service provider in step 3.

Note:

The identity provider list can be used by the LECP to create a user identifier to be presented to the Principal. For example, the LECP could compare the list of the Principal's known identities (and the identities of the identity provider that provides those identities) against the list provided by the service provider and then only display the intersection.

If the LECP discovers a syntax error due to the service provider or cannot proceed any further for other reasons (for example, cannot resolve identity provider, cannot reach the identity provider, etc.), the LECP MUST return to the service provider a `<lib:AuthnResponse>` with a `<samlp:Status>` indicating the desired error element as defined in [LibertyProtSchema]. The `<lib:AuthnResponse>` containing the error status MUST be sent using a POST to the service provider's assertion consumer service URL obtained from the `<lib:AssertionConsumerServiceURL>` element of the `<lib:AuthnRequestEnvelope>`. The POST MUST be a form that contains the field LARES with the value being the `<lib:AuthnResponse>` protocol message as defined in [LibertyProtSchema], containing the `<samlp:Status>`. The `<lib:AuthnResponse>` MUST be encoded by applying a base64 transformation (refer to [RFC2045]) to the `<lib:AuthnResponse>` and all its elements.

3.2.4.2.4. Step 6: HTTP Response with <AuthnResponse>

In step 6, the identity provider responds to the `<lib:AuthnRequest>` by issuing an HTTP 200 OK response. The response MUST contain a single `<lib:AuthnResponseEnvelope>` in the body of a SOAP message with content as defined in [LibertyProtSchema].

The identity provider MUST include the Liberty-Enabled HTTP header following the same processing rules as defined in 3.2.5.1.

The Content-Type MUST be set to application/vnd.liberty-response+xml.

If the identity provider discovers a syntax error due to the service provider or LECP or cannot proceed any further for other reasons (for example, an unsupported Liberty version), the identity provider MUST return to the LECP a `<lib:AuthnResponseEnvelope>` containing a `<lib:AuthnResponse>` with a `<samlp:Status>` indicating the desired error element as defined in [LibertyProtSchema].

3.2.4.2.5. Step 7: Posting the Form Containing the <AuthnResponse>

In step 7, the LECP issues an HTTP POST of the `<lib:AuthnResponse>` that was received in the `<lib:AuthnResponseEnvelope>` SOAP response in step 6. The `<lib:AuthnResponse>` MUST be sent using a POST to the service provider's assertion consumer service URL identified by the `<lib:AssertionConsumerServiceURL>` element within the `<lib:AuthnResponseEnvelope>` *obtained from the identity provider in step 6*. The POST MUST be a form that contains the field `LARES` with the value being the `<lib:AuthnResponse>` protocol message as defined in [LibertyProtSchema]. The `<lib:AuthnResponse>` MUST be encoded by applying a base64 transformation (refer to [RFC2045]) to the `<lib:AuthnResponse>` and

all its elements. The service provider's assertion consumer service URL used as the target of the form POST MUST specify `https` as the URL scheme; if another scheme is specified, it MUST be treated as an error by the identity provider.

If the LECP discovers an error (for example, syntax error in identity provider response), the LECP MUST return to the service provider a `<lib:AuthnResponse>` with a `<samlp:Status>` indicating the appropriate error element as defined in [LibertyProtSchema]. The `<ProviderID>` in the `<lib:AuthnResponse>` MUST be set to *urn:liberty:iff:lecp*. The `<lib:AuthnResponse>` containing the error status MUST be sent using a POST to the service provider's assertion consumer service URL. The POST MUST be a form that contains the field named LARES with its value being the `<lib:AuthnResponse>` protocol message as defined in [LibertyProtSchema] with formatting as specified Section 3.1.2. Any `<lib:AuthnResponse>` messages created by the identity provider MUST NOT be sent to the service provider.

3.3. Register Name Identifier Profiles

This section defines the profiles by which a provider may register or change a name identifier for a Principal. This message exchange is optional. During federation, the identity provider supplies an opaque handle identifying the Principle. This is the `<lib:IDPProvidedNameIdentifier>`. If neither provider involved in the federation opts to register any other name identifier, then this initial `<lib:IDPProvidedNameIdentifier>` is to be used by both providers.

An identity provider may choose to register a new `<lib:IDPProvidedNameIdentifier>` at any time subsequent to federation, using this protocol. Additionally, a service provider may choose to register a `<lib:SPProvidedNameIdentifier>`, which it expects the identity provider to use (instead of the `<lib:IDPProvidedNameIdentifier>`) when communicating with it about the Principal.

Two profiles are specified: HTTP-Redirect-Based and SOAP/HTTP-based.

Either the identity or service provider may initiate the register name identifier protocol. The available profiles are defined in Section 3.3.1 and Section 3.3.2, and vary slightly based on whether the protocol was initiated by the identity or service provider:

- Register Name Identifier Initiated at Identity Provider
  - HTTP-Redirect-Based: Relies on an HTTP 302 redirect to communicate between the identity provider and the service provider.
  - SOAP/HTTP-Based: Relies on a SOAP call from the identity provider to the service provider.
- Register Name Identifier Initiated at Service Provider
  - HTTP-Redirect-Based: Relies on an HTTP 302 redirect to communicate between the service provider and the identity provider.
  - SOAP/HTTP-Based: Relies on a SOAP call from the service provider to the identity provider.

The interactions and processing rules for the SOAP/HTTP-based and HTTP-redirect-based profiles are essentially the same regardless of whether the profile was initiated at the service provider or at the identity provider, but the message flow directions are reversed.

The register name identifier profiles make use of the following metadata elements, as defined in [LibertyMetadata]:

- `RegisterNameIdentifierProtocolProfile`: The service provider's preferred register name identifier profile, which should be used by the identity provider when registering a new identifier. This would specify the URI based identifier for one of the IDP Initiated register name identifier profiles.

- `RegisterNameIdentifierServiceURL`: The URL used for user-agent-based Register Name Identifier Protocol profiles.

- `RegisterNameIdentifierServiceReturnURL`: The provider's redirecting URL for use after HTTP name registration has taken place.

- `SOAPEndpoint`: The SOAP endpoint location at the service provider or identity provider to which Liberty SOAP messages are sent.

3.3.1. Register Name Identifier Initiated at Identity Provider

An identity provider MAY change the `<lib:IDPProvidedNameIdentifier>` it has assigned a Principal and transmit that information to a service provider. The `<lib:IDPProvidedNameIdentifier>` MAY be changed without changing any federations. The reasons an identity provider may wish to change the name identifier for a Principal are implementation dependent, and thus outside the scope of this specification. Changing the `<lib:IDPProvidedNameIdentifier>` MAY be accomplished in either an HTTP-Redirect-Based or SOAP/HTTP mode.

3.3.1.1. HTTP-Redirect-Based Profile

A HTTP-redirect-based register name identifier profile cannot be self-initiated by an identity provider, but must be a triggered by a message, such as an `<lib:AuthnRequest>`. We note that we do not normatively specify when and how the identity provider can initiate this profile - that is left to the discretion of the identity provider. As an example, it may be triggered by a message, such as an `<lib:AuthnRequest>`. When the identity provider decides to initiate the profile in this case, it will insert this profile between the `AuthnRequest/AuthnResponse` transactions.

The HTTP-redirect-based profile relies on using HTTP 302 redirects to communicate register name identifier messages from the identity provider to the service provider. The HTTP-Redirect Register Name Identifier Profile (Figure 6) illustrates this transaction.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://projectliberty.org/profiles/rni-idp-http`

This URI identifier MUST be specified in the service provider metadata element `RegisterNameIdentifierProtocolProfile` when the service provider intends to indicate to the identity provider a preference for receiving register name identifier messages via an HTTP 302 redirect.

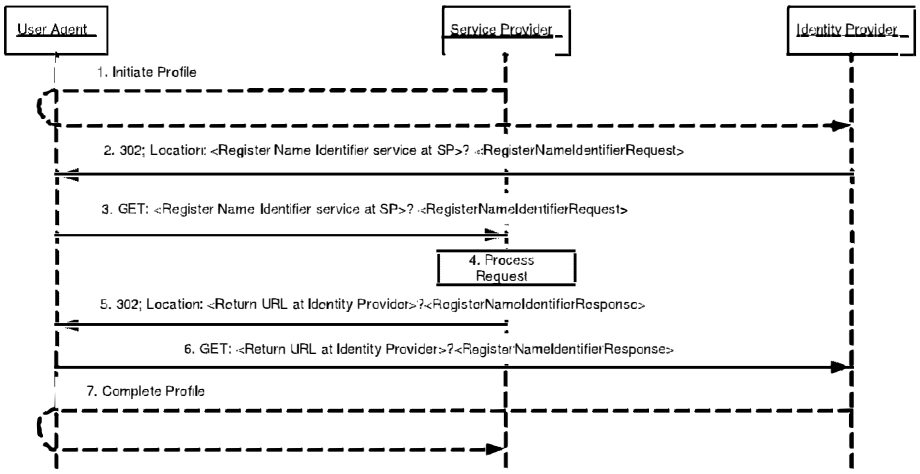


Figure 6. Register Name Identifier Profile.

In an example scenario, the service provider makes an `<lib:AuthnRequest>` to the identity provider for authentication of the Principal's User Agent (step 1). The identity provider effects an `<lib:IDPProvidedNameIdentifier>` change in the service provider via a URL redirection. The profile is as follows:

3.3.1.1.1. Step 1: Initiate Profile

This interaction is not normatively specified as part of the profile, but shown for illustrative purposes.

3.3.1.1.2. Step 2: Redirecting to the Service Provider Register Name Identifier Service

In step 2, the identity provider redirects the user agent to the register name identifier service at the service provider.

The redirection MUST adhere to the following rules:

- The Location HTTP header MUST be set to the service provider's register name identifier service URL.
- The service provider's register name identifier service URL MUST specify `https` as the URL scheme; if another scheme is specified, the identity provider MUST NOT redirect to the service provider.
- The Location HTTP header MUST include a `<query>` component containing the `<lib:RegisterNameIdentifierRequest>` protocol message as defined in [LibertyProtSchema] with formatting as specified in Section 3.1.2.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Service Provider Register Name Identifier service URL>?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

`<Service Provider Register Name Identifier service URL>`

This element provides the host name, port number, and path components of the register name identifier service URL at the service provider.

`<query>= ...<URL-encoded RegisterNameIdentifierRequest>...`

The <query> component MUST contain a single register name identifier request.

3.3.1.1.3. Step 3: Accessing the Service Provider Register Name Identifier Service

In step 3, the user agent accesses the service provider's register name identifier service URL with the `<lib:RegisterNameIdentifierRequest>` information attached to the URL fulfilling the redirect request.

3.3.1.1.4. Step 4: Processing the Register Name Identifier Request

In step 4, the service provider MUST process the `<lib:RegisterNameIdentifierRequest>` according to the rules defined in [LibertyProtSchema].

The service provider MAY remove the old name identifier after registering the new name identifier.

3.3.1.1.5. Step 5: Redirecting to the Identity Provider return URL with the Register Name Identifier Response

In step 5, the service provider's register name identifier service responds and redirects the user agent back to identity provider using a return URL location specified in the RegisterNameIdentifierServiceReturnURL metadata element. If the URL-encoded `<lib: RegisterNameIdentifierRequest>` message received in step 3 contains a parameter named RelayState, then the service provider MUST include a `<query>` component containing the same RelayState parameter and its value in its response to the identity provider.

The redirection MUST adhere to the following rules:

- The Location HTTP header MUST be set to the identity providers return URL specified in the RegisterNameIdentifierServiceReturnURL metadata element.
- The identity provider's return URL MUST specify `https` as the URL scheme; if another scheme is specified, the service provider MUST NOT redirect to the identity provider.
- The Location HTTP header MUST include a `<query>` component containing the `<lib:RegisterNameIdentifierResponse>` protocol message as defined in [LibertyProtSchema] with formatting as specified in Section 3.1.2.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Identity Provider Service Return URL >?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

`<Identity Provider Service Return URL>`

This element provides the host name, port number, and path components of the return URL at the identity provider.

`<query>= ...<URL-encoded RegisterNameIdentifierResponse>...`

The `<query>` component MUST contain a single register name identifier response. The `<URL-encoded RegisterNameIdentifierResponse>` component MUST contain the identical RelayState parameter and its value that was received in the URL-encoded register name identifier message obtained in step 3. If no RelayState parameter was provided in the step 3 message, then a RelayState parameter MUST NOT be specified in the `<URL-encoded RegisterNameIdentifierResponse>`.

3.3.1.1.6. Step 6: Accessing the Identity Provider return URL with the Register Name Identifier Response

In step 6, the user agent accesses the identity provider's return URL location fulfilling the redirect request.

3.3.1.1.7. Step 7: Complete profile

This concludes the initial sequence, which triggered the initiation of this profile.

3.3.1.2. SOAP/HTTP-Based Profile

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://projectliberty.org/profiles/rni-idp-soap`

This URI identifier MUST be specified in the service provider metadata element RegisterNameIdentifierProtocolProfile when the service provider intends to indicate to the identity provider a preference for receiving register name identifier messages via SOAP over HTTP.

The steps involved in the SOAP/HTTP-based profile MUST utilize the SOAP binding for Liberty as defined in Section 2.1. See Figure 7.

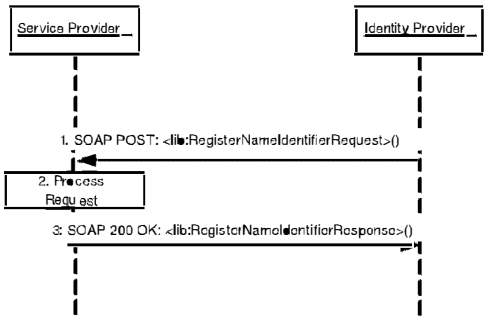


Figure 7. SOAP/HTTP-based profile for registering name identifiers

3.3.1.2.1. Step 1 Initiate Profile

In step 1, the identity provider sends a <lib:RegisterNameIdentifierRequest> protocol message to the service provider's SOAP endpoint specifying <lib:SPProvidedNameIdentifier>, <lib:IDPProvidedNameIdentifier>, and <lib:OldProvidedNameIdentifier> as defined in [LibertyProtSchema]. The <lib:SPProvidedNameIdentifier> will only contain a value if the service provider has previously used the register name identifier profile.

3.3.1.2.2. Step 2: Process Request

Service provider records new <lib:IDPProvidedNameIdentifier>.

3.3.1.2.3. Step 3: Response to Register Name Identifier

The service provider, after successfully registering the new <lib:IDPProvidedNameIdentifier> provided by the identity provider, MUST respond with a <lib:RegisterNameIdentifierResponse> according to the processing rules defined in [LibertyProtSchema].

3.3.2. Register Name Identifier Initiated at Service Provider

A service provider may register, or change a <lib:SPProvidedNameIdentifier> which is a name identifier it expects the identity provider to use when communicating with it about the Principal. Until it registers a <lib:SPProvidedNameIdentifier>, an identity provider will continue to use the current <lib:IDPProvidedNameIdentifier> when referring to the Principal.

3.3.2.1. HTTP-Redirect-Based Profile

The HTTP-redirect-based profile relies on the use of an HTTP 302 redirect to communicate a register name identifier message from the service provider to the identity provider.

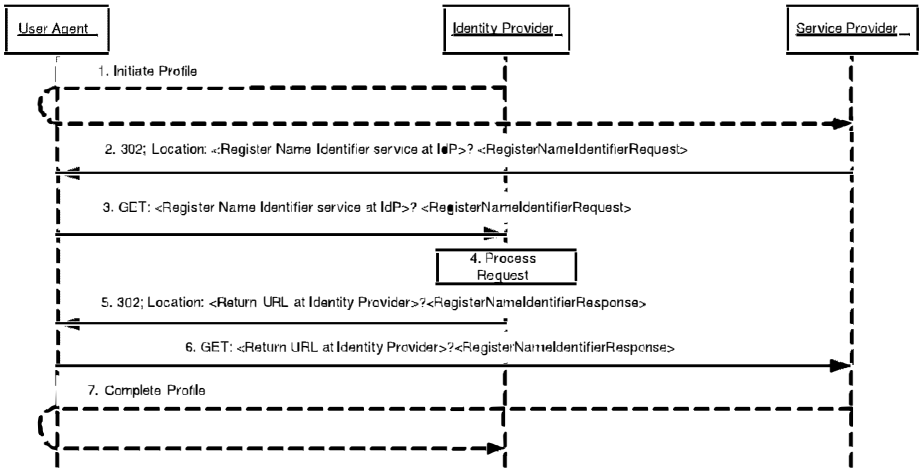


Figure 8. SP-Initiated Register Name Identifier Profile.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: http://projectliberty.org/profiles/rni-sp-http

A HTTP-redirect-based register name identifier profile can be self-initiated by a service provider to change the <lib:SPProvidedNameIdentifier;>. This does not normatively specify when and how the service provider can initiate this profile; that is left to the discretion of the service provider. The HTTP-redirect-based profile relies on

using HTTP 302 redirects to communicate register name identifier messages from the service provider to the identity provider. The service provider effects a `<lib:SPProvicedNameIdentifier>` change in the identity provider via a URL redirection. For a discussion of the interactions and processing steps, refer to Section 3.3.1.1. When reviewing that profile, interchange all references to service provider and identity provider in the interaction diagram and processing steps 2-6. See Figure 8. Note that in step 4 the old SPProvidedNameIdentifier SHOULD be removed at the IdP.

3.3.2.2. SOAP/HTTP-Based Profile

The SOAP/HTTP-based profile relies on using SOAP over HTTP to communicate register name identifier messages from the service provider to the identity provider. For a discussion of the interactions and processing steps, refer to Section 3.3.1.2. When reviewing that profile, interchange all references to service provider and identity provider in the interaction diagram and processing steps. See Figure 9.

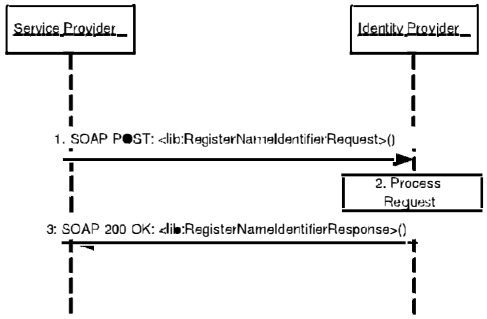


Figure 9. SP-Initiated SOAP/HTTP-based profile for registering name identifiers

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://prcjectliberty.org/profiles/rni-sp-soap`

In step 1, the service provider sends a `<lib:RegisterNameIdentifierRequest>` protocol message to the identity provider's SOAP endpoint specifying `<lib:SPProvidedNameIdentifier>`, `<lib:IDPProvidedNameIdentifier>`, and `<lib:OldProvidedNameIdentifier>` as defined in [LibertyProtSchema]. The `<lib:OldProvidedNameIdentifier>` will only contain a value if the service provider has previously used the register name identifier profile.

3.4. Identity Federation Termination Notification Profiles

The Liberty identity federation termination notification profiles specify how service providers and identity providers are notified of federation termination (also known as defederation).

Note:

Other means of federation termination are possible, such as federation expiration and termination of business agreements between service providers and identity providers. These means of federation termination are outside the scope of this specification.

Identity federation termination can be initiated at either the identity provider or the service provider. The Principal SHOULD have been authenticated by the provider at which identity federation termination is being initiated. The available profiles are defined in Section 3.4.1 and Section 3.4.2, depending on whether the identity federation termination notification process was initiated at the identity provider or service provider:

- Federation Termination Notification Initiated at Identity Provider

- HTTP-Redirect-Based: Relies on an HTTP 302 redirect to communicate between the identity provider and the service provider.
- SOAP/HTTP-Based: Relies on a SOAP call from the identity provider to the service provider.

- Federation Termination Notification Initiated at Service Provider
  - HTTP-Redirect-Based: Relies on an HTTP 302 redirect to communicate between the service provider and the identity provider.
  - SOAP/HTTP-Based: Relies on a SOAP call from the service provider to the identity provider.

The interactions and processing rules for the SOAP/HTTP-based and HTTP-redirect-based profiles are essentially the same regardless of whether federation termination notification was initiated at the service provider or at the identity provider.

The identity federation termination notification profiles make use of the following metadata elements, as defined in [LibertyMetadata]:

- `FederationTerminationServiceURL` - The URL at the service provider or identity provider to which identity federation termination notifications are sent. It is documented in these profiles as "federation termination service URL."
- `FederationTerminationServiceReturnURL` - The URL used by the service provider or identity provider when redirecting the user agent at the end of the federation termination notification profile process.
- `FederationTerminationNotificationProtocolProfile` - Used by the identity provider to determine which federation termination notification profile MUST be used when communicating with the service provider.
- `SOAPEndpoint` - The SOAP endpoint location at the service provider or identity provider to which Liberty SOAP messages are sent.

3.4.1. Federation Termination Notification Initiated at Identity Provider

The profiles in Section 3.4.1.1 and Section 3.4.1.2 are specific to identity federation termination when initiated at the identity provider. Effectively, when using these profiles, the identity provider is stating to the service provider that it will no longer provide the Principal's identity information to the service provider and that the identity provider will no longer respond to any requests by the service provider on behalf of the Principal.

3.4.1.1. HTTP-Redirect-Based Profile

The HTTP-redirect-based profile relies on using HTTP 302 redirect to communicate federation termination notification messages from the identity provider to the service provider. See Figure 10.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://projectliberty.org/profiles/fedterm-idp-http`

This URI identifier MUST be specified in the service provider metadata element FederationTerminationNotification-ProtocolProfile when the service provider intends to indicate to the identity provider a preference for receiving federation termination notifications via an HTTP 302 redirect.

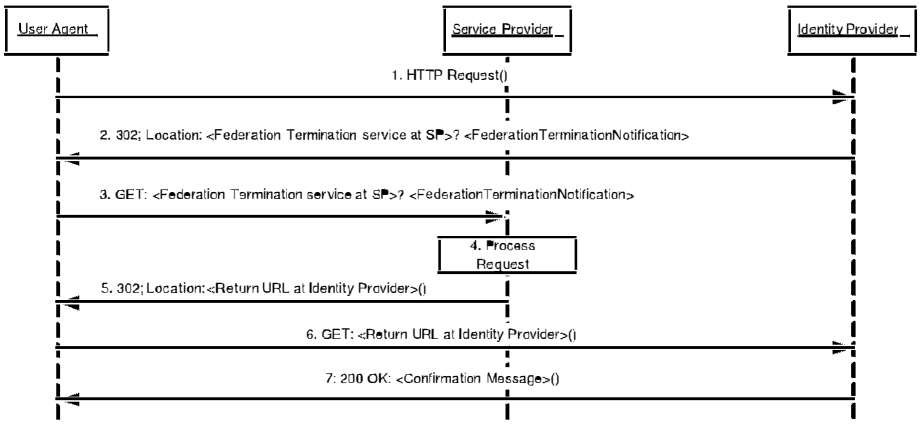


Figure 10. HTTP-redirect-based profile for federation termination

This profile description assumes the following preconditions:

- The Principal's identity at the service provider is federated with his/her identity at the identity provider.
- The Principal has requested to the identity provider that the federation be terminated.
- The Principal has authenticated with the identity provider.

3.4.1.1.1. Step 1: Accessing the Federation Termination Service

In step 1, the user agent accesses the identity federation termination service URL at the identity provider specifying the service provider with which identity federation termination should occur. How the service provider is specified is implementation-dependent and, as such, is out of the scope of this specification.

3.4.1.1.2. Step 2: Redirecting to the Service Provider

In step 2, the identity provider's federation termination service URL responds and redirects the user agent to the federation termination service at the service provider.

The redirection MUST adhere to the following rules:

- The Location HTTP header MUST be set to the service provider's federation termination service URL.
- The service provider's federation termination service URL MUST specify https as the URI scheme; if another scheme is specified, the identity provider MUST NOT redirect to the service provider.
- The Location HTTP header MUST include a `<query>` component containing the `<lib:FederationTerminationNotification>` protocol message as defined in [LibertyProtSchema] with formatting as specified in Section 3.1.2.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Service Provider Federation Termination service URL>?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

`<Service Provider Federation Termination service URL>`

This element provides the host name, port number, and path components of the federation termination service URL at the service provider.

`<query>= ...<URL-encoded FederationTerminationNotification>...`

The `<query>` component MUST contain a single terminate federation request.

3.4.1.1.3. Step 3: Accessing the Service Provider Federation Termination Service

In step 3, the user agent accesses the service provider's federation termination service URL with the `<lib:FederationTerminationNotification>` information attached to the URL fulfilling the redirect request.

3.4.1.1.4. Step 4: Processing the Notification

In step 4, the service provider MUST process the `<lib:FederationTerminationNotification>` according to the rules defined in [LibertyProtSchema].

The service provider MAY remove any locally stored references to the name identifier it received from the identity provider in the `<lib:FederationTerminationNotification>`.

3.4.1.1.5. Step 5: Redirecting to the Identity Provider Return URL

In step 5, the service provider's federation termination service responds and redirects the user agent back to identity provider using a return URL location specified in the FederationTerminationServiceReturnURL metadata element. If the URL-encoded `<lib:FederationTerminationNotification>` message received in step 3 contains a parameter named RelayState, then the service provider MUST include a `<query>` component containing the same RelayState parameter and its value in its response to the identity provider.

No success or failure message should be conveyed in this HTTP redirect. The sole purpose of this redirect is to return the user agent to the identity provider where the federation termination process began.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Identity Provider Service Return URL >?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

`<Identity Provider Service Return URL>`

This element provides the components of the return URL at the identity provider.

1544 `<query>= . . .RelayState=<. . .>`

1545 The `<query>` component MUST contain the identical RelayState parameter and its value that was received in the
1546 URL-encoded federation termination message obtained in step 3. If no RelayState parameter was provided in the step
1547 3 message, then a RelayState parameter MUST NOT be specified in the `<query>` component.

1548 #### 3.4.1.1.6. Step 6: Accessing the Identity Provider Return URL

1549 In step 6, the user agent accesses the identity provider's return URL location fulfilling the redirect request.

1550 #### 3.4.1.1.7. Step 7: Confirmation

1551 In step 7, the user agent is sent an HTTP response that confirms the requested action of identity federation termination
1552 with the specific service provider.

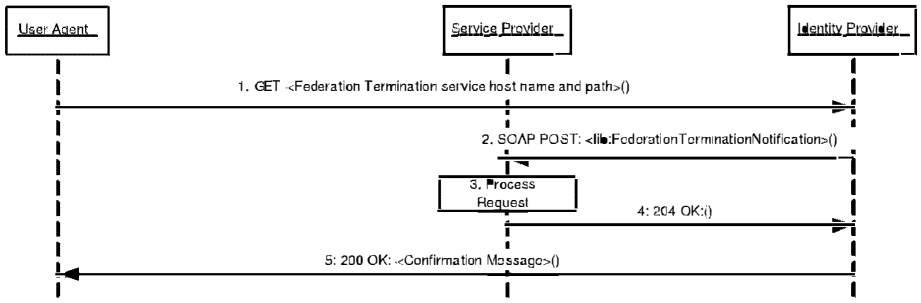
1553 ### 3.4.1.2. SOAP/HTTP-Based Profile

1554 The SOAP/HTTP-based profile relies on using asynchronous SOAP over HTTP to communicate federation termination
1555 notification messages from the identity provider to the service provider. See Figure 11.

1556 The following URI-based identifier MUST be used when referencing this specific profile:

1557 URI: `http://projectliberty.org/profiles/fedterm-idp-soap`

1558 This URI identifier MUST be specified in the service provider metadata element FederationTerminationNotification-
1559 ProtocolProfile when the service provider intends to indicate to the identity provider a preference for receiving feder-
1560 ation termination notifications via SOAP over HTTP.



1561

1562 **Figure 11. SOAP/HTTP-based profile for federation termination**

1563 This profile description assumes the following preconditions:

1564 • The Principal's identity at the service provider is federated with his/her identity at the identity provider.

1565 • The Principal has authenticated with the identity provider.

1566 • The Principal has requested that the identity provider terminate the federation.

3.4.1.2.1. Step 1: Accessing the Federation Termination Service

In step 1, the user agent accesses the identity federation termination service URL at the identity provider specifying the service provider for with which identity federation termination should occur. How the service provider is specified is implementation-dependent and, as such, is out of the scope of this specification.

3.4.1.2.2. Step 2: Notification of Federation Termination

In step 2, the identity provider sends an asynchronous SOAP over HTTP notification message to the service provider's SOAP endpoint. The SOAP message MUST contain exactly one `<lib:FederationTerminationNotification>` element in the SOAP body and adhere to the construction rules defined in [LibertyProtSchema].

If a SOAP fault occurs, the identity provider SHOULD employ best effort to resolve the fault condition and resend the federation termination notification message to the service provider.

3.4.1.2.3. Step 3: Processing the Notification

In step 3, the service provider MUST process the `<lib:FederationTerminationNotification>` according to the rules defined in [LibertyProtSchema].

The service provider MAY remove any locally stored references to the name identifier it received from the identity provider in the `<lib:FederationTerminationNotification>`.

3.4.1.2.4. Step 4: Responding to the Notification

In step 4, the service provider MUST respond to the `<lib:FederationTerminationNotification>` with an HTTP 204 OK response.

3.4.1.2.5. Step 5: Confirmation

In step 5, the user agent is sent an HTTP response that confirms the requested action of identity federation termination with the specific service provider.

3.4.2. Federation Termination Notification Initiated at Service Provider

The profiles in Section 3.4.2.1 and Section 3.4.2.2 are specific to identity federation termination notification when initiated by a Principal at the service provider. Effectively, when using this profile, the service provider is stating to the identity provider that the Principal has requested that the identity provider no longer provide the Principal's identity information to the service provider and that service provider will no longer ask the identity provider to do anything on the behalf of the Principal.

It is RECOMMENDED that the service provider, after initiating or receiving a federation termination notification, invalidate the local session for the Principal that was authenticated at the identity provider with which federation has been terminated. If the Principal was locally authenticated at the service provider, the service provider MAY continue to maintain a local session for the Principal. If the Principal wants to engage in a single sign-on session with identity provider again, the service provider MUST first federate with identity provider the given Principal.

3.4.2.1. HTTP-Redirect-Based Profile

The HTTP-redirect-based profile relies on the use of an HTTP 302 redirect to communicate a federation termination notification message from the service provider to the identity provider. For a discussion of the interactions and processing steps, refer to Section 3.4.1.1. When reviewing that profile, interchange all references to service provider and identity provider in the interaction diagram and processing steps.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://projectliberty.org/profiles/fedterm-sp-http`

This URI identifier is really only meant for service provider consumption and as such is not needed in any provider metadata.

3.4.2.2. SOAP/HTTP-Based Profile

The SOAP/HTTP-based profile relies on using asynchronous SOAP over HTTP to communicate federation termination notification messages from the service provider to the identity provider. For a discussion of the interactions and processing steps, refer to 3.4.1.2. When reviewing that profile, interchange all references to service provider and identity provider in the interaction diagram and processing steps.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://prcjectliberty.org/profiles/fedterm-sp-s●ap`

This URI identifier is really only meant for service provider consumption and as such is not needed in any provider metadata.

3.5. Single Logout Profiles

The single logout profiles synchronize session logout functionality across all sessions that were authenticated by a particular identity provider. The single logout can be initiated at either the identity provider or the service provider. In either case, the identity provider will then communicate a logout request to each service provider with which it has established a session for the Principal. The negotiation of which single logout profile the identity provider uses to communicate with each service provider is based upon the SingleLogoutProtocolProfile provider metadata element defined in [LibertyProtSchema].

The available profiles are defined in Section 3.5.1 and Section 3.5.2, depending on whether the single logout is initiated at the identity provider or service provider:

- *Single Logout Initiated at Identity Provider*
  - HTTP-Based: Relies on using either HTTP 302 redirects or HTTP GET requests to communicate logout requests from an identity provider to the service providers.
  - SOAP/HTTP-Based: Relies on SOAP over HTTP messaging to communicate logout requests from an identity provider to the service providers.
- *Single Logout Initiated at Service Provider*
  - HTTP-Redirect-Based: Relies on an HTTP 302 redirect to communicate a logout request with the identity provider.
  - SOAP/HTTP-Based: Relies on SOAP over HTTP messaging to communicate a logout request from a service provider to an identity provider.

The single logout profiles make use of the following metadata elements, as defined in [LibertyMetadata]:

- `SingleLogoutServiceURL` — The URL at the service provider or identity provider to which single logout requests are sent. It is described in these profiles as "single logout service URL."

- `SingleLogoutServiceReturnURL`— The URL used by the service provider when redirecting the user agent to the identity provider at the end of the single logout profile process.
- `SingleLogoutProtocolProfile` — Used by the identity provider to determine which single logout request profile MUST be used when communicating with the service provider.
- `SOAPEndpoint` — The SOAP endpoint location at the service provider or identity provider to which Liberty SOAP messages are sent.

3.5.1. Single Logout Initiated at Identity Provider

The profiles in 3.5.1.1 through 3.5.1.2 are specific to a single logout when initiated by a user agent at the identity provider.

3.5.1.1. HTTP-Based Profile

The HTTP-based profile defines two possible implementations that an identity provider may use. The first implementation relies on using HTTP 302 redirects, while the second uses HTTP GET requests. The choice of implementation is entirely dependent upon the type of user experience the identity provider provides.

The following URI-based identifier MUST be used when referencing either implementation for this specific profile:

URI: `http://projectliberty.org/profiles/slo-idp-http`

This URI identifier MUST be specified in the service provider metadata element SingleLogoutProtocolProfile when the service provider intends to indicate to the identity provider a preference for receiving logout requests via either an HTTP redirect or an HTTP GET.

3.5.1.1.1. HTTP-Redirect Implementation

The HTTP-Redirect implementation uses HTTP 302 redirects to communicate a logout request to each service provider for which the identity provider has provided authentication assertions during the Principal's current session if the service provider indicated a preference to receive logout requests via the HTTP based profile. See Figure 12.

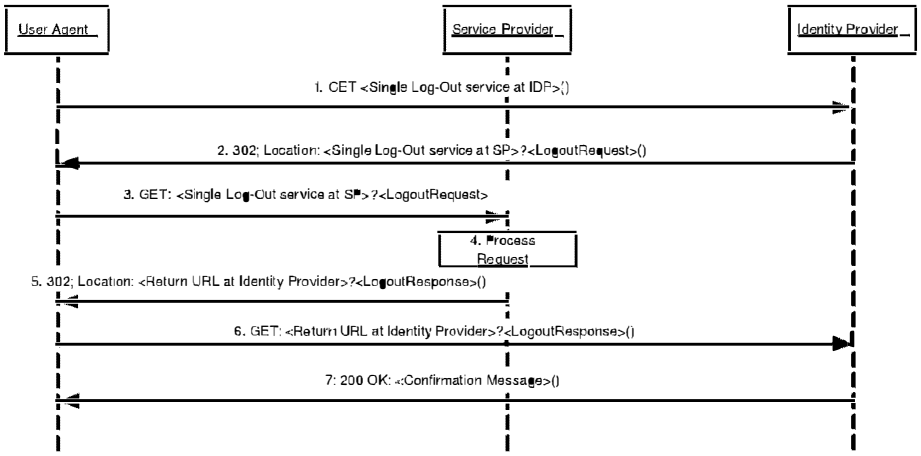


Figure 12. HTTP-Redirect implementation for single logout initiated at identity provider

Notes:

Steps 2 through 6 may be an iterative process for requesting logouts by each service provider that has been issued authentication assertions during the Principal's current session and has indicated a preference to receive logout requests via the HTTP based profile.

[RFC2616] indicates a client should detect infinite redirection loops because such loops generate network traffic for each redirection. This requirement was introduced because previous versions of the specification recommended a maximum of five redirections. Content developers should be aware that some clients might implement such a fixed limitation.

3.5.1.1.1.1. Step 1: Accessing the Single Logout Service at the Identity Provider

In step 1, the user agent accesses the single logout service URL at the identity provider indicating that all service providers for which this identity provider has provided authentication assertions during the Principal's current session must be notified of session termination.

3.5.1.1.1.2. Step 2: Redirecting to the Single Logout Service at the Service Provider

In step 2, the identity provider's single logout service responds and redirects the user agent to the single logout service URL at each service provider for which the identity provider has provided an authentication assertion during the Principal's current session with the identity provider.

The redirections MUST adhere to the following rules:

- The Location HTTP header MUST be set to the service provider's single logout service URL.
- The service provider's single logout service URL MUST specify https as the URL scheme; if another scheme is specified, the identity provider MUST NOT redirect to the service provider.
- The Location HTTP header MUST include a `<query>` component containing the `<lib:LogoutRequest>` protocol message as defined in [LibertyProtSchema] with formatting as specified in 3.1.2.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Service Provider Single Log-Out service URL>?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

`<Service Provider Single Log-Out service URL>`

This element provides the host name, port number, and path components of the single logout service URL at the service provider.

`<query>= ...<URL-encoded LogoutRequest>...`

The `<query>` MUST contain a single logout request.

3.5.1.1.1.3. Step 3: Accessing the Service Provider Single Logout Service

In step 3, the user agent accesses the service provider's single logout service URL with the `<lib:LogoutRequest>` information attached to the URL fulfilling the redirect request.

3.5.1.1.1.4. Step 4: Processing the Request

In step 4, the service provider MUST process the `<lib:LogoutRequest>` according to the rules defined in [LibertyProtSchema].

The service provider MUST invalidate the session(s) of the Principal referred to in the name identifier it received from the identity provider in the <lib:LogoutRequest>.

3.5.1.1.1.5. Step 5: Redirecting to the Identity Provider Return URL

In step 5, the service provider's single logout service responds and redirects the user agent back to the identity provider using the return URL location obtained from the SingleLogoutServiceReturnURL metadata element. If the URL-encoded <lib:LogoutRequest> message received in step 3 contains a parameter named RelayState, then the service provider MUST include a <query> component containing the same RelayState parameter and its value in its response to the identity provider.

The purpose of this redirect is to return the user agent to the identity provider so that the single logout process may continue in the same fashion with other service providers.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Identity Provider Service Return URL>?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

<Identity Provider Service Return URL>

This element provides the host name, port number, and path components of the return URL at the identity provider.

<query>= ...<URL-encoded LogoutResponse>

The <query> component MUST contain a single logout response. The <URL-encoded LogoutResponse> MUST contain the identical RelayState parameter and its value that was received in the URL-encoded logout request message obtained in step 3. If no RelayState parameter was provided in the step 3 message, then a RelayState parameter MUST NOT be specified in the <URL-encoded LogoutResponse>.

3.5.1.1.1.6. Step 6: Accessing the Identity Provider Return URL

In step 6, the user agent accesses the identity provider's return URL location fulfilling the redirect request.

3.5.1.1.1.7. Step 7: Confirmation

In step 7, the user agent is sent an HTTP response that confirms the requested action of a single logout has been completed.

3.5.1.1.2. HTTP-GET Implementation

The HTTP-GET implementation uses HTTP GET requests to communicate logout requests to each service provider for which the identity provider has provided authentication during the Principal's current session if the service provider indicated a preference to receive logout requests via the HTTP based profile. See Figure 13.

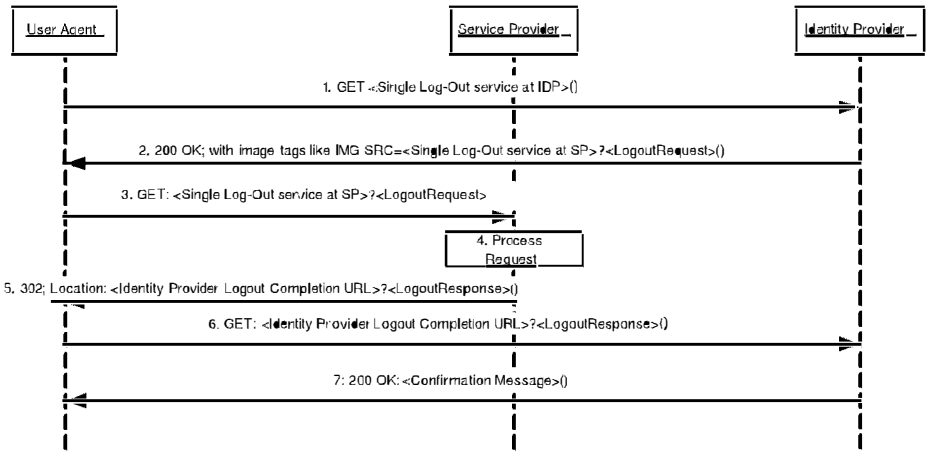


Figure 13. HTTP-GET implementation for single logout initiated at identity provider

Note:

Steps 3 through 7 may be an iterative process for requesting logout of each service provider that has been issued authentication assertions during the Principal's current session and has indicated a preference to receive logout requests via the HTTP based profile.

3.5.1.1.2.1. Step 1: Accessing the Single Logout Service at the Identity Provider

In step 1, the user agent accesses the single logout service URL at the identity provider indicating that all service providers for which this identity provider has provided authentication assertions during the Principal's current session must be notified of session termination and requested to logout the Principal.

3.5.1.1.2.2. Step 2: HTML Page Returned to User Agent with Image Tags

In step 2, the identity provider's single logout service responds with an HTML page that includes image tags referencing the logout service URL for each of the service providers for which the identity provider has provided an authentication assertion during the Principal's current session. The list of image tags MUST be sent in a standard HTTP 200 response to the user agent.

The image tag loads on the HTML page MUST adhere to the following rules:

- The SRC attribute MUST be set to the specific service provider's single logout service URL.
- The service provider's single logout service URL MUST specify https as the URL scheme.
- The service provider's single logout service URL MUST include a `<query>` component containing the `<lib:LogoutRequest>` protocol message as defined in [LibertyProtSchema] with formatting as specified in 3.1.2.

3.5.1.1.2.3. Step 3: Accessing the Service Provider Single Logout Service

In step 3, the user agent, as a result of each image load, accesses the service provider's single logout service URL with <lib:LogoutRequest> information attached to the URL. This step may occur multiple times if the HTTP response includes multiple image tag statements (one for each service provider that has been issued authentication assertions during the Principal's current session).

3.5.1.1.2.4. Step 4: Processing the Request

In step 4, the service provider MUST process the <lib:LogoutRequest> according to the rules defined in [LibertyProtSchema].

The service provider MUST invalidate the session of the Principal referred to in the name identifier it received from the identity provider in the <lib:LogoutRequest>.

3.5.1.1.2.5. Step 5: Redirecting to the Identity Provider Logout Completion URL

In step 5, the service provider's single logout service responds and redirects the image load back to the identity provider's logout completion URL. This location will typically point to an image that will be loaded by the user agent to indicate that the logout is complete (for example, a checkmark).

The logout completion URL is obtained from the SingleLogoutServiceReturnURL metadata element.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Identity Provider Logout Completion URL>?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

<Identity Provider Logout Completion URL>

This element provides the host name, port number, and path components of the identity provider logout completion URL at the identity provider.

<query>=...<URL-encoded LogoutResponse>

The <query> component MUST contain a single logout response. The <URL-encoded LogoutResponse> component MUST contain the identical RelayState parameter and its value that was received in the URL-encoded logout request message obtained in step 3. If no RelayState parameter was provided in step 3 then a RelayState message MUST NOT be specified in the <URL-encoded LogoutResponse>.

3.5.1.1.2.6. Step 6: Accessing the Identity Provider Logout Completion URL

In step 6, the user agent accesses the identity provider's logout completion URL fulfilling the redirect request.

3.5.1.1.2.7. Step 7: Confirmation

In step 7, the user agent is sent an HTTP response that confirms the requested action of a single logout has been completed.

Note:

One method for seamlessly returning the user agent back to the identity provider is for the HTML page generated in step 2 to include a script that runs when the page is completely loaded (all logouts completed) that will initiate the redirect to the identity provider.

3.5.1.2. SOAP/HTTP-Based Profile

The SOAP/HTTP-based profile uses SOAP over HTTP messaging to communicate a logout request to each service provider for which the identity provider has provided authentication assertions during the Principal's current session if the service provider indicated a preference to receive logout request via the SOAP/HTTP-based profile. See Figure 14.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://projectliberty.org/profiles/slo-idp-soap`

This URI identifier MUST be specified in the service provider metadata element SingleLogoutProtocolProfile when the service provider intends to indicate to the identity provider a preference for receiving logout requests via SOAP over HTTP.

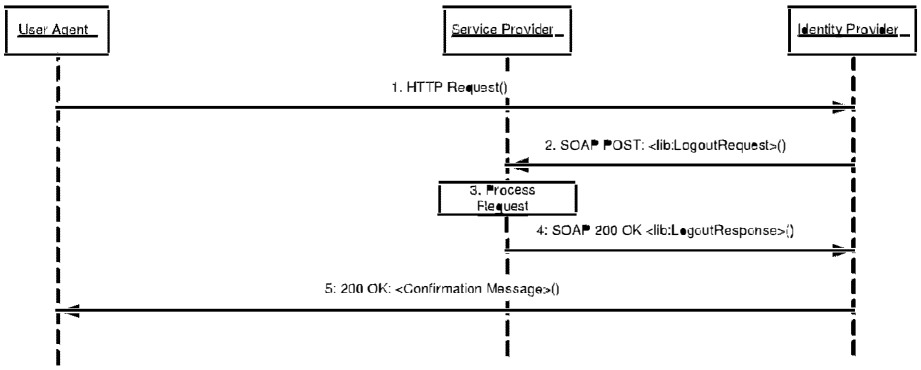


Figure 14. SOAP/HTTP-based profile for single logout initiated at identity provider

Note:

Steps 2 through 4 may be an iterative process for each service provider that has been issued authentication assertions during the Principal's current session and has indicated a preference to receive logout requests via the SOAP/HTTP message profile.

3.5.1.2.1. Step 1: Accessing the Single Logout Service

In step 1, the user agent accesses the single logout service URL at the identity provider via an HTTP request.

3.5.1.2.2. Step 2: Logout Request

In step 2, the identity provider sends a SOAP over HTTP request to the SOAP endpoint of each service provider for which it provided authentication assertions during the Principal's current session. The SOAP message MUST contain exactly one `<lib:LogoutRequest>` element in the SOAP body and adhere to the construction rules defined in [LibertyProtSchema].

If a SOAP fault occurs, the identity provider SHOULD employ best efforts to resolve the fault condition and resend the single logout request to the service provider.

3.5.1.2.3. Step 3: Processing the Logout Request

In step 3, the service provider MUST process the `<lib:LogoutRequest>` according to the rules defined in [LibertyProtSchema].

The service provider MUST invalidate the session for the Principal specified by the name identifier provided by the identity provider in the `<lib:LogoutRequest>`.

3.5.1.2.4. Step 4: Responding to the Request

In step 4, the service provider MUST respond to the `<lib:LogoutRequest>` with a SOAP 200 OK `<lib:LogoutResponse>` message.

3.5.1.2.5. Step 5: Confirmation

In step 5, the user agent is sent an HTTP response that confirms the requested action of single logout has completed.

3.5.2. Single Logout Initiated at Service Provider

The profiles in Section 3.5.2.1 and Section 3.5.2.2 are specific to the Principal' initiation of the single logout request process at the service provider.

3.5.2.1. HTTP-Based Profile

The HTTP-based profile relies on using an HTTP 302 redirect to communicate a logout request with the identity provider. The identity provider will then communicate a logout request to each service provider with which it has established a session for the Principal using the service provider' preferred profile for logout request from the identity provider (see Section 3.5.1). See Figure 15.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://projectliberty.org/profiles/slo-sp-http`

This URI identifier is intended for service provider consumption and is not needed in provider metadata.

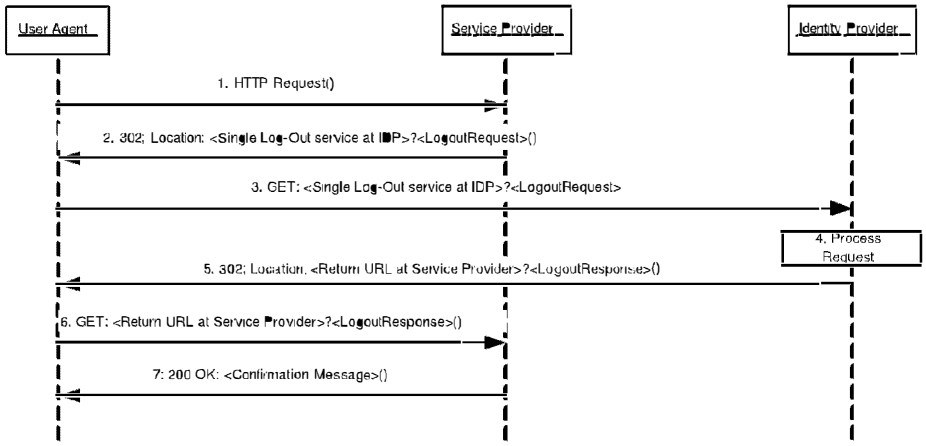


Figure 15. HTTP-redirect-based profile for single logout initiated at service provider

Note:

Step 4 may involve an iterative process by the identity provider to implement the preferred profile for logout requests for each service provider that has been issued authentication assertions during the Principal's current session.

3.5.2.1.1. Step 1: Accessing the Single Logout Service at the Service Provider

In step 1, the user agent accesses the single logout service URL at the service provider indicating that session logout is desired at the associated identity provider and all service providers for which this identity provider has provided

authentication assertions during the Principal's current session. If a current session exists for the Principal at the service provider, it is RECOMMENDED that the service provider terminate that session prior to step 2.

3.5.2.1.2. Step 2: Redirecting to the Single Logout Service at the Identity Provider

In step 2, the service provider's single logout service responds and redirects the user agent to the single logout service URL at the identity provider.

The redirection MUST adhere to the following rules:

- The Location HTTP header MUST be set to the identity provider's single logout service URL.
- The identity provider's single logout service URL MUST specify https as the URL scheme; if another scheme is specified, the service provider MUST NOT redirect to the identity provider.
- The Location HTTP header MUST include a `<query>` component containing the `<lib:LogoutRequest>` protocol message as defined in [LibertyProtSchema] with formatting as specified in 3.1.2.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Identity Provider single log-out service URL>?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

`<Identity Provider single log-out service URL>`

This element provides the host name, port number, and path components of the single logout service URL at the identity provider.

`<query>= ...<URL-encoded LogoutRequest>...`

The `<query>` MUST contain a single logout request.

3.5.2.1.3. Step 3: Accessing the Identity Provider Single Logout Service

In step 3, the user agent accesses the identity provider's single logout service URL with the `<lib:LogoutRequest>` information attached to the URL fulfilling the redirect request.

3.5.2.1.4. Step 4: Processing the Request

In step 4, the identity provider MUST process the `<lib:LogoutRequest>` according to the rules defined in [LibertyProtSchema].

Each service provider for which the identity provider has provided authentication assertions during the Principal's current session MUST be notified via the service provider's preferred profile for logout request from the identity provider (see Section 3.5.1).

The identity provider's current session with the Principal MUST be terminated, and no more authentication assertions for the Principal are to be given to service providers.

3.5.2.1.5. Step 5: Redirecting to the Service Provider Return URL

In step 5, the identity provider's single logout service responds and redirects the user agent back to service provider using the return URL location obtained from the SingleLogoutServiceReturnURL metadata element. If the URL-encoded <lib:LogoutRequest> message received in step 3 contains a parameter named RelayState, then the identity provider MUST include a <query> component containing the same RelayState parameter and its value in its response to the service provider.

The purpose of this redirect is to return the user agent to the service provider.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<Service Provider Return Service URL>?<query>
<other HTTP 1.0 or 1.1 components>
```

where:

<Service Provider Service Return URL>

This element provides the host name, port number, and path components of the return URL location at the service provider.

<query>- ...<URL-encoded LogoutResponse>

The <query> component MUST contain a single logout response. The <URL-encoded LogoutResponse> component MUST contain the identical RelayState parameter and its value that was received in the URL-encoded logout request message obtained in step 3. If no RelayState parameter was provided in the step 3 message, then a RelayState parameter MUST NOT be specified in the <URL-encoded LogoutResponse>.

3.5.2.1.6. Step 6: Accessing the Service Provider Return URL

In step 6, the user agent accesses the service provider's return URL location fulfilling the redirect request.

3.5.2.1.7. Step 7: Confirmation

In step 7, the user agent is sent an HTTP response that confirms the requested action of a single logout has been completed.

3.5.2.2. SOAP/HTTP-Based Profile

The SOAP/HTTP-based profile relies on using SOAP over HTTP messages to communicate a logout request to the identity provider. The identity provider will then communicate a logout request to each service provider it has established a session with for the Principal via the service provider' preferred profile for logout requests from the identity provider (see Section 3.5.1). See Figure 16.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: http://prcjectliberty.org/profiles/slo-sp-soap

This URI identifier is intended for service provider consumption and is not needed in provider metadata.

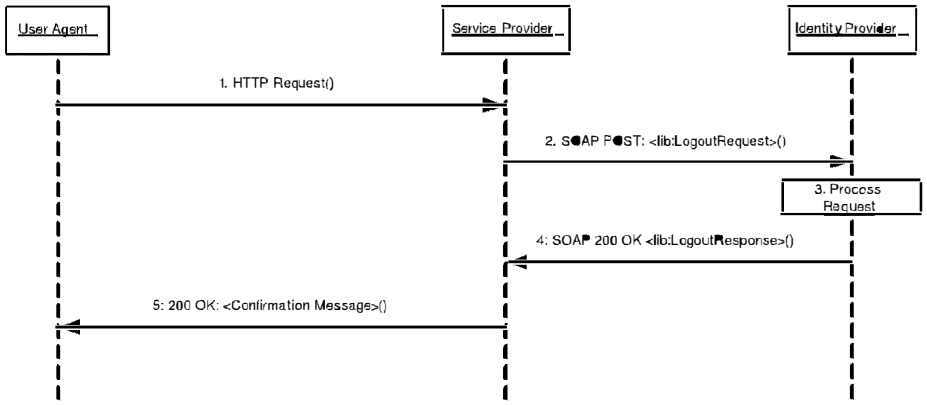


Figure 16. SOAP/HTTP-based profile for single logout initiated at service provider

Note:

Step 3 may involve an iterative process by the identity provider to implement the preferred profile for logout requests for each service provider that has been issued authentication assertions during the Principal's current session.

3.5.2.2.1. Step 1: Accessing Single Logout Service

In step 1, the user agent accesses the single logout service URL at the service provider via an HTTP request.

3.5.2.2.2. Step 2: Logout Request

In step 2, the service provider sends a SOAP over HTTP request to the identity provider's SOAP endpoint. The SOAP message MUST contain exactly one `<lib:LogoutRequest>` element in the SOAP body and adhere to the construction rules as defined in [LibertyProtSchema].

If a SOAP fault occurs, the service provider SHOULD employ best efforts to resolve the fault condition and resend the single logout request to the identity provider.

3.5.2.2.3. Step 3: Processing the Logout Request

In step 3, the identity provider MUST process the `<lib:LogoutRequest>` according to the rules defined in [LibertyProtSchema].

Each service provider for which the identity provider has provided authentication assertions during the Principal's current session MUST be requested to logout the Principal via the service provider's preferred profile for logout requests from the identity provider. If the identity provider determines that one or more of service providers to which it has provided assertions regarding this Principal do not support the SOAP profiles for the single logout, the identity provider MUST return a `<lib:LogoutResponse>` containing a status code of `<lib:UnsupportedProfile>`. The service provider MUST then re-submit its LogoutRequest via the HTTP profile described above.

Note:

If the identity provider is proxying authentication, based on authentication assertions from a second (proxied) identity provider (see Dynamic Proxying of Identity Providers in [LibertyProtSchema]), then the identity provider MUST follow these processing steps, *prior* to attempting to propagate the Single Logout request to service providers for which they have provided authentication assertions (as described above):

1. If the identity provider determines that the proxied identity provider does not support the SOAP/HTTP profile of Single Logout, it MUST respond to the requesting service provider with a `<lib:LogoutResponse>` containing a status code of `<lib:UnsupportedProfile>`.

2. If the identity provider is able to proceed with SOAP/HTTP-based Single Logout, then it MUST initiate the SP-initiated SOAP/HTTP profile of Single Logout described in Section 3.5.2.2, *acting in the role of service provider* toward the proxied identity provider. The proxied identity provider (in processing the SP-initiated SLO request) may determine that some service provider for which it provided authentication assertions does not support the SOAP/HTTP profile of Single Logout, and might thus return a `<lib:UnsupportedProfile>` status response. If this situation occurs, the proxying identity provider MUST return a `<lib:UnsupportedProfile>` status response to the requesting service provider.

The identity provider's current session with the Principal MUST be terminated, and no more authentication assertions for the Principal are to be given to service providers.

3.5.2.2.4. Step 4: Responding to the Logout Request

In step 4, the identity provider MUST respond to the `<lib:LogoutRequest>` with a SOAP 200 OK `<lib:LogoutResponse>` message.

3.5.2.2.5. Step 5: Confirmation

In step 5, the user agent is sent an HTTP response that confirms the requested action of single logout was completed.

3.6. Identity Provider Introduction

This section defines the profiles by which a service provider discovers which identity providers a Principal is using. In identity federation networks having more than one identity provider, service providers need a means to discover which identity providers a Principal uses. The introduction profile relies on a cookie that is written in a domain that is common between identity providers and service providers in an identity federation network. The domain that the identity federation network predetermines for a deployment is known as the common domain in this specification, and the cookie containing the list of identity providers is known as the common domain cookie.

Implementation of this profile is OPTIONAL. Whether identity providers and service providers implement this profile is a policy and deployment issue outside the scope of this specification. Also, which entities host web servers in the common domain is a deployment issue and is outside the scope of this specification.

3.6.1. Common Domain Cookie

The name of the cookie MUST be _liberty_idp. The format of the cookie content MUST be a list of base64-encoded (see [RFC2045]) identity provider succinct IDs separated by a single white space character. The identity provider IDs MUST adhere to the creation rules as defined in Section 3.2.2.2. The identity provider ID is a metadata element, as defined in [LibertyMetadata].

The common domain cookie writing service SHOULD append the identity provider ID to the list. If the identity provider ID is already present in the list, it MAY remove and append it when authentication of the Principal occurs. The intent is that the most recently established identity provider session is the last one in the list.

The cookie MUST be set with no Path prefix or a Path prefix of "/". The Domain MUST be set to ".[common-domain]" where [common-domain] is the common domain established within the identity federation network for use with the introduction protocol. The cookie MUST be marked as Secure.

The cookie SHOULD be URL-encoded.

Cookie syntax should be in accordance with [RFC2965] or [NetscapeCookie].

The cookie MAY be either session or persistent. This choice may be made within an identity federation network, but should apply uniformly to all providers in the network (see [LibertyImplGuide]) for more details on cookies).

3.6.2. Setting the Common Domain Cookie

After the identity provider authenticates a Principal, it MAY set the common domain cookie. The means by which the identity provider sets the cookie are implementation-specific so long as the cookie is successfully set with the parameters given above. One possible implementation strategy follows and should be considered non-normative. The identity provider may:

- Have previously established a DNS and IP alias for itself in the common domain
- Redirect the user agent to itself using the DNS alias using a URL specifying "`https`" as the URL scheme. The structure of the URL is private to the implementation and may include session information needed to identify the user-agent.
- Set the cookie on the redirected user agent using the parameters specified above.
- Redirect the user agent back to itself, or, if appropriate, to the service provider.

3.6.3. Obtaining the Common Domain Cookie

When a service provider needs to discover which identity providers the Principal uses, it invokes a protocol exchange designed to present the common domain cookie to the service provider after it is read by an HTTP server in the common domain.

If the HTTP server in the common domain is operated by the service provider, the service provider MAY redirect the user agent to an identity provider's intersite transfer service for an optimized single sign-on process.

The specific means by which the service provider reads the cookie are implementation-specific so long as it is able to cause the user agent to present cookies that have been set with the parameters given in Section 3.6.1. One possible implementation strategy is described as follows and should be considered non-normative. Additionally, it may be sub-optimal for some applications.

- Have previously established a DNS and IP alias for itself in the common domain
- Redirect the user agent to itself using the DNS alias using a URL specifying "`https`" as the URL scheme. The structure of the URL is private to the implementation and may include session information needed to identify the user-agent.
- Set the cookie on the redirected user agent using the parameters specified above
- Redirect the user agent back to itself, or, if appropriate, to the service provider.

3.7. NameIdentifier Mapping Profile

The NameIdentifier mapping profile specifies how a service provider may obtain a NameIdentifier for a Principal it has federated in the "namespace" of a different service provider, by querying an identity provider that has federated the Principal with both service providers. This NameIdentifier may be used to obtain additional information about a Principal from a SAML authority, or used for other non-specific purposes. In most cases, the encryption profile in the following section will be used to obfuscate and time-limit this identifier to restrict its use.

The NameIdentifier mapping profile makes use of the following metadata elements, as defined in [LibertyMetadata]:

- `NameIdentifierMappingProtocolProfile` - A URI indicating the profile of this protocol supported by the identity provider.
- `SOAPEndpoint` - The SOAP endpoint location at the identity provider to which Liberty SOAP messages are sent.

3.7.1. SOAP-based NameIdentifier Mapping

The SOAP-based profile relies on a SOAP request and response to query for and return the NameIdentifier. A requesting service provider issues a SOAP request to an identity provider, requesting a different NameIdentifier for a named Principal in the namespace of a service provider or affiliation group. This NameIdentifier may then be used to query another Liberty provider offering SAML services for additional information about the named Principal. See Figure 17.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `http://prcjectliberty.org/profiles/nim-sp-http`

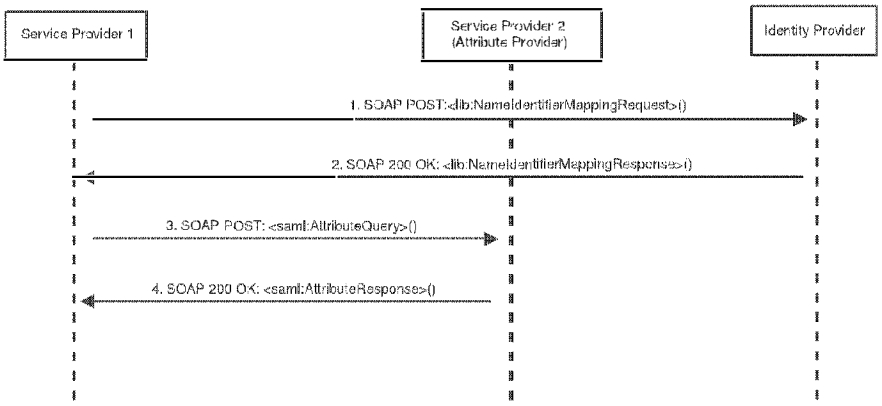


Figure 17. SOAP-based profile for name identifier mapping

3.7.1.1. Step 1: Issuance and processing of Request

In step 1, the service provider sends a SOAP over HTTP request to the SOAP endpoint of the identity provider it is querying. The SOAP message MUST contain exactly one `<lib:NameIdentifierMappingRequest>` element in the SOAP body and adhere to the construction rules defined in [LibertyProtSchema].

The identity provider MUST process the `<lib:NameIdentifierMappingRequest>` according to the rules defined in [LibertyProtSchema]. The identity provider is NOT required to honor the request for a mapped NameIdentifier, but it MUST respond to the request with an appropriate status.

3.7.1.2. Step 2: Responding to the Request

In step 3, the identity provider MUST respond to the `<lib:NameIdentifierMappingRequest>` with a SOAP 200 OK `<lib:NameIdentifierMappingResponse>` message.

3.7.1.3. Step 3: Requesting SAML attributes using a mapped NameIdentifier

Note: This step is not normatively specified by Liberty, and is shown only for illustrative purposes. The requesting service provider may use the mapped NameIdentifier of the Principal to issue a `<saml:AttributeQuery>`. This MUST adhere to the rules specified in [SAMLCore11]

3.7.1.4. Step 4: Returning a <saml:AttributeStatement>

Note: This step is not normatively specified by Liberty, and is shown only for illustrative purposes. A service provider receiving a `<saml:AttributeQuery>` may return a `<saml:AttributeStatement>`. This action MUST conform to the rules specified in [SAMLCore11].

3.7.1.5. Security Considerations

In addition to the usual considerations relating to Liberty and SAML protocols (see [SAMLCore11]), an identity provider SHOULD encrypt or otherwise obfuscate the NameIdentifier returned to the requesting service provider, so that it is opaque to the requester. A way of accomplishing this is described in the next section.

Because the identifier gives the receiving provider a persistent way of referencing the principal, it should only be returned subject to the policies set by the principal or other authorized party.

3.8. NameIdentifier Encryption Profile

The Liberty NameIdentifier encryption profile allows a principal's NameIdentifier to be encrypted such that only the identity or service provider possessing the decryption key can deduce the identity of the principal when the NameIdentifier is included in a SAML or Liberty protocol message. The identifier is encrypted in such a fashion that it is a different value when requested by different providers or multiple times, reducing the chance for correlation of the encrypted value across multiple logical transactions.

The NameIdentifier encryption profile make use of the following metadata element, as defined in [LibertyMetadata]:

- `KeyDescriptor` - Defines a public key to use when wrapping the keys used in encrypting data for a provider (the key-encrypting key)

3.8.1. XML Encryption-based NameIdentifier Encryption

The XML Encryption-based profile relies on the use of [xmlenc-core] to format and encode the resulting encrypted identifier and possibly the wrapped encryption key.

The following URI-based identifier MUST be used when referencing this specific profile:

URI: `urn:liberty:iff:nameid:encrypted`

3.8.1.1. Step 1: Encrypting and encoding a NameIdentifier value.

The encrypting provider first transforms the original `<saml:NameIdentifier>` element into a `<EncryptableNameIdentifier>` element, which is an extension of the original element. The `NameQualifier`, `IssueInstant`, and `Nonce` attributes are set as defined by [LibertyProtSchema].

If not already generated for the target provider, an encryption key is generated and is then itself encrypted with the key specified in the target provider's `<KeyDescriptor>` metadata element with a `use` attribute of `encryption`, or with a predefined key exchanged out of band. The wrapped encryption key is placed into a `<xenc:EncryptedKey>` element.

If the symmetric encryption key is not included, because it has been exchanged out of band, and/or is being reused, then the encrypting provider MUST include additional information in the `<xenc:EncryptedData>` element that indicates to the target provider which decryption key to use in decrypting the identifier. This information MUST be sufficient to identify the key to use without the target knowing the encrypting provider's identity before decryption occurs.

The encryption key is then applied to the `<EncryptableNameIdentifier>` element, producing an `<xenc:EncryptedData>` element with a `Type` of `http://www.w3.org/2001/04/xmlenc#Element`.

The resulting `<xenc:EncryptedData>` element, and optionally the `<xenc:EncryptedKey>` element, are then enclosed in an `<EncryptedNameIdentifier>` element. The element is base-64 encoded and the result is placed into a `<saml:NameIdentifier>` whose `Format` attribute MUST be `urn:liberty:iff:nameid:encrypted`.

3.8.1.2. Step 2: Decoding and decrypting a NameIdentifier value.

The decrypting provider first decodes the base-64 encoded data and recovers the `<EncryptedNameIdentifier>` element.

The `<xenc:EncryptedData>` and optional `<xenc:EncryptedKey>` elements are then used to recover the symmetric encryption key and algorithm and decrypt the `<EncryptableNameIdentifier>` element.

The provider can then examine the attributes to determine the identity federation to which the name identifier applies.

3.8.2. Security Considerations

The profile is designed to meet the needs of providers in addressing the security considerations of other profiles, such as the NameIdentifier Mapping Profile in the previous section. To insure the integrity of this profile, either symmetric encryption keys MUST NOT be reused, or if they are, then symmetric encryption keys MUST be reused between different principals federated with a given provider and MUST NOT be reused between different providers. It is RECOMMENDED that symmetric encryption keys, if reused, be renewed periodically. Furthermore, reuse of keys REQUIRES that a chaining mode with a unique initialization vector generated per encryption be used.

4. Security Considerations

4.1. Introduction

This section describes security considerations associated with Liberty protocols for identity federation, single sign-on, federation termination, and single logout.

Liberty protocols, schemas, bindings, and profiles inherit and use extensively the SAML protocols. Therefore, the security considerations published along with the SAML specification have direct relevance (see [SAMLCore11], [SAMLBind11], and [SAMLSec]). Throughout this section if, for any reason, a specific consideration or counter-measure does not apply or differs, notice of this fact is made; and a description of alternatives is supplied, where possible.

4.2. General Requirements

4.2.1. Security of SSL and TLS

SSL and TLS utilize a suite of possible cipher suites. The security of the SSL or TLS session depends on the chosen cipher suite. An entity (that is, a user agent, service provider, or identity provider) that terminates an SSL or TLS connection needs to offer (or accept) suitable cipher suites during the handshake. The following list of TLS 1.0 cipher suites (or their SSL 3.0 equivalent) is recommended.

- TLS_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

The above list is not exhaustive. The recommended cipher suites are among the most commonly used. Note: New cipher suites are added as they are standardized and should be considered for inclusion if they have sufficiently strong security properties. For example, it is anticipated that the AES-based cipher suites being standardized in the IETF will be widely adopted and deployed.

4.2.2. Security Implementation

The suitable implementation of security protocols is necessary to maintain the security of a system, including

- Secure random or pseudo-random number generation
- Secure storage

4.3. Classification of Threats

4.3.1. Threat Model

For an analysis of threat classifications, an Internet threat model has been used. In other words, the threat model assumes that intermediary and end-systems participating in Liberty protocol exchanges have not been compromised. However, where possible, the consequences and containment properties of a compromised system entity are described and countermeasures are suggested to bolster the security posture so that the exposure from a security breach is minimized.

Given the nature of the Internet, the assumption is made that deployment is across the global Internet and, therefore, crosses multiple administrative boundaries. Thus, an assumption is also made that the adversary has the capacity to engage in both passive and active attacks (see Section 4.3.3).

4.3.2. Rogue and Spurious Entities

Attackers may be classified based on their capabilities and the roles that they play in launching attacks on a Liberty system as follows:

- **Rogue Entities:** Entities that misuse their privileges. The rogue actors may be Principals, user agents, service providers, or identity providers. A rogue Principal is a legitimate participant who attempts to escalate its privileges or masquerade as another system Principal. A rogue user agent may, for instance, misuse the relationships between its associated Principals and an identity provider to launch certain attacks. Similarly, a rogue service provider may be able to exploit the relationship that it has either with a Principal or with an identity provider to launch certain attacks.

- **Spurious Entities:** Entities that masquerade as a legitimate entity or are completely unknown to the system. The spurious actors include Principals, user agents (i.e., user agents without associated legitimate Liberty Principals), service providers, or identity providers. A spurious service provider may, for instance, pretend to be a service provider that has a legitimate relationship with an identity provider. Similarly, a spurious Principal may be one that pretends to be a legitimate Principal that has a relationship with either a service provider or an identity provider.

4.3.3. Active and Passive Attackers

Both rogue and spurious entities may launch active or passive attacks on the system. In a passive attack the attacker does not inject traffic or modify traffic in any way. Such an attacker usually passively monitors the traffic flow, and the information that is obtained in that flow may be used at a later time. An active attacker, on the other hand, is capable of modifying existing traffic as well as injecting new traffic into the system.

4.3.4. Scenarios

The following scenarios describe possible attacks:

- **Collusion: The secret cooperation between two or more Liberty entities to launch an attack, for example,**

  - Collusion between Principal and service provider
  - Collusion between Principal and identity provider
  - Collusion between identity provider and service provider

- Collusion among two or more Principals
- Collusion between two or more service providers
- Collusion between two or more identity providers

- **Denial-of-Service Attacks:** The prevention of authorized access to a system resource or the delaying of system operations and functions.
- **Man-in-the-Middle Attacks:** A form of active wiretapping attack in which the attacker intercepts and selectively modifies communicated data to masquerade as one or more of the entities involved in a communication association.
- **Replay Attacks:** An attack in which a valid data transmission is maliciously or fraudulently repeated, either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack.
- **Session Hijacking:** A form of active wiretapping in which the attacker seizes control of a previously established communication association.

4.4. Threat Scenarios and Countermeasures

In this section, threats that may apply to all the Liberty profiles are considered first. Threats that are specific to individual profiles are then considered. In each discussion the threat is described as well as the countermeasures that exist in the profile or the additional countermeasures that may be implemented to mitigate the threat.

4.4.1. Common Threats for All Profiles

Threat: Request messages sent in cleartext

Description: Most profile protocol exchanges do not mandate that all exchanges commence over a secure communication channel. This lack of transport security potentially exposes requests and responses to both passive and active attacks.

One obvious manifestation is when the initial contact is not over a secure transport and the Liberty profile begins to exchange messages by carrying the request message back to the user agent in the location header of a redirect.

Another such manifestation could be a request or response message which carries a URI that may be resolved on a subsequent exchange, for instance lib:AuthnContextClassRef. If this URI were to specify a less or insecure transport, then the exchange may be vulnerable to the types of attacks described above.

Countermeasure: Ensure that points of entry to Liberty protocol exchanges utilize the https URL `<scheme>` and that all interactions for that profile consistently exchange messages over `https`.

Threat: Malicious redirects into identity or service provider targets

Description: A spurious entity could issue a redirect to a user agent so that the user agent would access a resource that disrupts single sign-on. For example, an attacker could redirect the user agent to a logout resource of a service provider causing the Principal to be logged out of all existing authentication sessions.

Countermeasure: Access to resources that produce side effects could be specified with a transient qualifier that must correspond to the current authentication session. Alternatively, a confirmation dialog could be interposed that relies on a transient qualifier with similar semantics.

Threat: Relay state tampering or fabrication

Description: Some of the messages may carry a <lib:RelayState> element, which is recommended to be integrity-protected by the producer and optionally confidentiality-protected. If these practices are not followed, an adversary could trigger unwanted side effects. In addition, by not confidentiality-protecting the value of this element, a legitimate system entity could inadvertently expose information to the identity provider or a passive attacker.

Countermeasure: Follow the recommended practice of confidentiality- and integrity-protecting the <lib:RelayState> data. Note: Because the value of this element is both produced and consumed by the same system entity, symmetric cryptographic primitives could be utilized.

4.4.2. Single Sign-On and Federation

4.4.2.1. Common Interactions for All Single Sign-On and Federation Profiles

Threat: <lib:AuthnRequest> sent over insecure channel

Description: It is recommended that the initial exchange to access the intersite transfer service be conducted over a TLS-secured transport. Not following this recommendation can expose the exchange to both passive and active attacks.

Countermeasure: Deploy the intersite transfer service under an https scheme.

Threat: Unsigned <lib:AuthnRequest> message

Description: The signature element of an <lib:AuthnRequest> is optional and thus the absence of the signature could pose a threat to the identity provider or even the targeted service provider. For example, a spurious system entity could generate an unsigned <lio:AuthnRequest> and redirect the user agent to the identity provider. The identity provider must then consume resources.

Countermeasure: Sign the <lib:AuthnRequest>. The IDP can also verify the identity of the Principal in the absence of a signed request.

Threat: Replay of an authentication assertion

Description: After obtaining a valid assertion from an identity provider, either legitimately or surreptitiously, the entity replays the assertion to the Service at a later time. A digital signature must cover the entire assertion, thus elements within the assertion cannot be corrupted without detection during the mandatory verification step. However, it is possible to fabricate an <lib:AuthnResponse> with the valid assertion.

Countermeasure: The issuer should sign <lib:AuthnResponse> messages. Signing binds the <samlp:IssueInstant> of the response message to the assertion it contains. This binding accords the relying party the opportunity to temporally judge the response. Additionally, a valid signature over the response binds the <samlp:InResponseTo> element to the corresponding <lib:AuthnRequest>. (Specifying a short period that the authentication assertion can be relied upon will minimize, but not mitigate this threat. Binding the <lib:AssertionId> to the request <samlp:InResponseTo> element may also be handy.)

Threat: Fabricated <lib:AuthnResponse> denial of service

Description: An attacker captures the <samlp:RequestID> sent in an <lib:AuthnRequest> message by a service provider to an identity provider, and sends several spurious <lib:AuthnResponse> messages to the service provider with the same <samlp:InResponseTo>. Because the <samlp:InResponseTo> matches a <samlp:RequestID> that the service provider had used, the service provider goes through the process of validating the signature in the message. Thus, it is subject to a denial of service attack.

Countermeasure: A secure communication channel should be established before transferring requests and responses.

Threat: Collusion between two Principals

Description: After getting an artifact or <lib:AuthnResponse> in step 6 (see Section 3.2.1), a legitimate Principal A could pass this artifact or <lib:AuthnResponse> on to another Principal, B. Principal B is now able to use the artifact or <lib:AuthnResponse>, while the actual authentication happened via Principal A.

Countermeasure: Implementations where this threat is a concern MUST use the <saml:AuthenticationLocality> in the authentication statement. The IP address that Principal B uses would be different from the IP address within the <saml:AuthenticationLocality>. This countermeasure may not suffice when the user agent is behind a firewall or proxy server. IP spoofing may also circumvent this countermeasure.

Threat: Stolen artifact and subsequent Principal impersonation

Description: See Section 4.1.1.9.1 in [SAMLBind11]

Countermeasure: Identity providers MUST enforce a policy of one-time retrieval of the assertion corresponding to an artifact so that a stolen artifact can be used only once. Implementations where this threat is a concern MUST use the <saml:AuthenticationLocality> in the authentication statement. The IP address of a spurious user agent that attempts to use the stolen artifact would be different from IP address within the <saml:AuthenticationLocality>. The service provider may then be able to detect that the IP addresses differ. This countermeasure may not suffice when the user agent is behind a firewall or proxy server. IP address spoofing may also circumvent this countermeasure.

Threat: Stolen assertion and subsequent Principal impersonation

Description: See Section 4.1.1.9.1 in [SAMLBind11]

Countermeasure: Refer to the previous threat for requirements.

Threat: Rogue service provider uses artifact or assertion to impersonate Principal at a different service provider

Description: Because the <lib:AuthnResponse> contains the <lib:ProviderID>, this threat is not possible.

Countermeasure: None

Threat: Rogue identity provider impersonates Principal at a service provider

Description: Because the Principal trusts the identity provider, it is assumed that the identity provider does not misuse the Principal's trust.

Countermeasure: None

Threat: Identity provider modifies Principal during a session with a service provider

Description: A service provider whose session has exceeded the <ReauthenticateOnOrAfter> time must contact the Identity provider to get a new assertion. The new assertion might be for a different identity.

Countermeasure: Service providers should continue to follow assertion processing rules to ensure that the subject of any assertions received is actually the user for which the assertion is needed.

Threat: Rogue user attempts to impersonate currently logged-in legitimate Principal and thereby gain access to protected resources.

Description: Once a Principal is successfully logged into an identity provider, subsequent <AuthnRequest> messages from different service providers concerning that Principal will not necessarily cause the Principal to be reauthenticated. Principals must, however, be authenticated unless the identity provider can determine that an <AuthnRequest> is associated not only with the Principal's identity, but also with a validly authenticated identity provider session for that Principal.

Countermeasure: In implementations where this threat is a concern, identity providers MUST maintain state information concerning active sessions, and MUST validate the correspondence between an <AuthnRequest> and

an active session before issuing an <AuthnResponse> without first authenticating the Principal. Cookies posted by identity providers MAY be used to support this validation process, though Liberty does not mandate a cookie-based approach.

4.4.2.2. Liberty-Enabled Client and Proxy Profile

Threat: Intercepted <lib:AuthnRequestEnvelope> and <lib:AuthnResponse> and subsequent Principal impersonation.

Description: A spurious system entity can interject itself as a man-in-the-middle (MITM) between the user agent (LECP) and a legitimate service provider, where it acts in the service provider role in interactions with the LECP, and in the user agent role in interactions with the legitimate service provider. In this way, as a first step, the MITM is able to intercept the service provider's <lib:AuthnRequestEnvelope> (step 3 of Section 3.2.4) and substitute any URL of its choosing for the <lib:AssertionConsumerServiceURL> value before forwarding the <lib:AuthnRequestEnvelope> on to the LECP. Typically, the MITM will insert a URL value that points back to itself. Then, if the LECP subsequently receives a <lib:AuthnResponseEnvelope> from the identity provider (step 6 in Section 3.2.4) and subsequently sends the contained <lib:AuthnResponse> to the <lib:AssertionConsumerServiceURL> received from the MITM, the MITM will be able to masquerade as the Principal at the legitimate service provider.

Countermeasure: The identity provider specifies to the LECP the address to which the LECP must send the <lib:AuthnResponse>. The <lib:AssertionConsumerServiceURL> in the <lib:AuthnResponseEnvelope> element is for this purpose. This URL value is among the metadata that identity and service providers must exchange in the process of establishing their operational relationship (see Section 3.1 and Section 3.1.3).

4.4.2.3. Federation

Threat: Collusion among service providers can violate privacy of the Principal

Description: When a group of service providers collude to share the <lib:IDPProvidedNameIdentifier> of a Principal, they can track and in general compromise the privacy of the principal. More generally, this threat exists for any common data (e.g. phone number) shared by rogue system entities.

Countermeasure: The <lib:IDPProvidedNameIdentifier> is required to be unique for each identity provider to service provider relationship. However, this requirement does not eliminate the threat when there are rogue participants under the Principal's identity federation. The only protection is for Principals to be cautious when they choose service providers and understand their privacy policies.

Threat: Poorly generated name identifiers may compromise privacy

Description: The federation protocol mandates that the <lib:NameIdentifier> elements be unique within a Principal's federated identities. The name identifiers exchanged are pseudonyms and, to maintain the privacy of the Principal, should be resistant to guessing or derivation attacks.

Countermeasure: Name identifiers should be constructed using pseudo-random values that have no discernable correspondence with the Principal's identifier (or name) used by the entity that generates the name identifier.

4.4.3. Name Registration

No known threats.

4.4.4. Federation Termination: HTTP-Redirect-Based Profile

Threat: Attacker can monitor and disrupt termination

Description: During the initial steps, a passive attacker can collect the <lib:FederationTerminationNotification> information when it is issued in the redirect. This threat is possible because the first and second steps are not required to use https as the URL scheme. An active attacker may be able to intercept and modify the message conveyed in step 2 because the digital signature only covers a portion of the message. This initial exchange also exposes the name identifier. Exposing these data poses a privacy threat.

Countermeasure: All exchanges should be conducted over a secure transport such as SSL or TLS.

4.4.5. Single Logout: HTTP-Redirect-Based Profile

Threat: Passive attacker can collect a Principal's name identifier

Description: During the initial steps, a passive attacker can collect the <lib:LogoutRequest> information when it is issued in the redirect. Exposing these data poses a privacy threat.

Countermeasure: All exchanges should be conducted over a secure transport such as SSL or TLS.

Threat: Unsigned <lio:LogoutRequest> message

Description: An Unsigned <lib:LogoutRequest> could be injected by a spurious system entity thus denying service to the Principal. Assuming that the NameIdentifier can be deduced or derived then it is conceivable that the user agent could be directed to deliver a fabricated <lio:LogoutRequest> message.

Countermeasure: Sign the <lib:LogoutRequest> message. The identity provider can also verify the identity of a Principal in the absence of a signed request.

4.4.6. Identity Provider Introduction

No known threats.

References

Normative

[HTML4] Raggett, D., Le Hors, A., Jacobs, I, eds. (24 December 1999). "HTML 4.01 Specification ," Recommendation, W3C *http://www.w3.org/TR/html401/ [August 2003].*

[LibertyAuthnContext] Madsen, Paul, eds. "Liberty ID-FF Authentication Context Specification," Version 1.2-errata-v1.0, Liberty Alliance Project (12 September 2004). *http://www.projectliberty.org/specs*

[LibertyGlossary] "Liberty Technical Glossary," Version 1.3-errata-v1.0, Liberty Alliance Project (12 Aug 2004). *http://www.projectliberty.org/specs* Hodges, Jeff, eds.

[LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 1.0-errata-v2.0, Liberty Alliance Project (12 September 2004). *http://www.projectliberty.org/specs*

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004). *http://www.projectliberty.org/specs*

[SAMLBind11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification, version 1.1, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/documents.php?wg_abbrev=security*

[SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification, version 1.1, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/documents.php?wg_abbrev=security*

[SAMLSec] McClaren, C., eds. (05 November 2002). "Security Considerations for the OASIS Security Assertion Markup Language (SAML)," Version 1.0, OASIS Standard, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/security/#documents*

[Schema1] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (May 2002). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlschema-1/*

[SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David , Kakivaya, Gopal, Layman, Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C Note (08 May 2000). *http://www.w3.org/TR/2000/NOTE-SOAP-20000508/*

[SSL] Frier, A., Karlton, P., Kocher, P., eds. (November 1996). Netscape Communications Corporation "The SSL 3.0 Protocol," *http://www.netscape.com/eng/ssl3/*

[RFC1750] Eastlake , D., Crocker, S., Schiller, J., eds. (December 1994). "Randomness Recommendations for Security ," RFC 1750, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc1750.txt [August 2003].*

[RFC1945] Berners-Lee, T., Fielding, R., Frystyk, H., eds. (May 1996). "Hypertext Transfer Protocol – HTTP/1.0 ," RFC 1945, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc1945.txt [August 2003].*

[RFC2045] Freed, N., Borenstein, N., eds. (November 1996). "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies ," RFC 2045., Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2045.txt [November 1996].*

[RFC2965] Kristol, D., Montulli, L., eds. (October 2000). "HTTP State Management Mechanism," RFC 2965., Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2965.txt [October 2000].*

[RFC2119] Bradner, S., eds. "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet Engineering Task Force (March 1997). *http://www.ietf.org/rfc/rfc2119.txt [March 1997].*

[RFC2246] Dierks, T., Allen, C., eds. (January 1999). "The TLS Protocol," Version 1.0 RFC 2246, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2246.txt [January 1999].*

[RFC2396] Berners-Lee, T., Fielding, R., Masinter, L., eds. (August 1998). "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2396.txt [August 1998].*

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June 1999). "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2616.txt [June 1999].*

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Stewart, L., eds. (June 1999). "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2617.txt [June 1999].*

[RFC3106] Eastlake , D., Goldstein, T., eds. (April 2001). "ECML v1.1: Field Specifications for E-Commerce ," RFC 3106, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc3106.txt [April 2001].*

[WML] "Wireless Markup Language Version 1.3 Specification," Version 1.3, Open Mobile Alliance *http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html*

[xmlenc-core] Eastlake, Donald, Reagle, Joseph, eds. (December 2002). "XML Encryption Syntax and Processing," W3C Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlenc-core/*

Informative

[LibertyImplGuide] "Liberty ID-FF Implementation Guidelines," Version 1.2, Liberty Alliance Project (18 April 2004). *http://www.projectliberty.org/specs* Thompson, Peter, Champagne, Darryl, eds.

[NetscapeCookie] eds. "Persistent Client State HTTP Cookies," Netscape *http://wp.netscape.com/newsref/std/cookie_spec.html*

EXHIBIT 5

Privacy and Security Best Practices

Version 2.0

November 12, 2003

Editor:

Christine Varney, Hogan & Hartson

Contributors:

Piper Cole, Sun Microsystems
William Duserick, Fidelity
Jill Lesser, AOL
Gary Podorowsky, Sony
Paule Sibieta, France Telecom
Charlotte Thornby, Sun Microsystems

Abstract:

Privacy and security are key concerns in the implementation of Liberty Alliance specifications. As such, the Liberty Alliance has and will continue to provide tools and guidance to implementing companies that enable them to build more secure, privacy-friendly identity-based services that can comply with local regulations and create a more trusted relationship with customers and partners. The following document highlights certain national privacy laws, fair information practices and implementation guidance for organizations using the Liberty Alliance specifications.

Contents

1. Executive Summary

The Liberty Alliance Project is a consortium of more than 150 organizations worldwide working together to create open, technical specifications for federated network identity. These specifications, which are available for any organization to download and incorporate into products and services, provide:

- Simplified sign-on capabilities using a federated network identity architecture that supports all current and emerging network access devices.
- Permissions-based attribute sharing to enable organizations to provide users with choice and control over the use and disclosure of their personal information.
- A commonly accepted platform and mechanism for building and managing identity-based web services based on open industry standards.

Because companies will implement Liberty's specifications in connection with their web-based offerings, privacy and security are key concerns. As such, Liberty has and will continue to provide tools and guidance to implementing companies to lead them to build more secure, privacy-friendly identity-based services that can be in compliance with local regulations and create a more trusted relationship with customers and partners.

The Liberty specifications will enable companies to adhere to information practices that comply with national privacy laws and regulations, or in the absence of such laws, industry best practices. It is however, important to note that Liberty, as a standards body, will not manage companies' compliance with those laws.

The following document highlights certain national privacy laws, fair information practices and implementation guidance for organizations using the Liberty Alliance specifications. Below is a brief summary of key points made in the paper, but the Alliance would encourage all parties to read through the entire document.

Liberty's Perspective on Privacy:

- The Liberty Alliance considers privacy and security of a Principal's personal information to be extremely important. This philosophy has driven many decisions crucial in the specification development process.
- Because the Liberty specifications represent a novel approach to account linking and data exchange, we believe it is important to identify implementation best practices to accompany the specifications.
- The Alliance also recommends that companies implementing any identity-related services or applications consult with local counsel to ensure that the services they provide comply with applicable privacy laws and regulations.

Privacy Laws and Fair Information Practices:

- There are several privacy laws, in varying states of development, enacted worldwide. This document highlights some of the more pertinent regulations in the U.S., Canada, Europe and other regions.
- This paper also describes certain practices and frameworks created by various organizations that address the use and disclosure of personal information. Topics addressed include adoption and implementation of privacy policies, notice and disclosure, choice and consent, data quality, security safeguards and accountability.

Liberty Alliance Privacy Recommendations:

- In addition to current laws and organizational guidelines, the Liberty Alliance offers in this paper a baseline set of fair information practices for any entity using the Liberty specifications.
- Liberty-enabled providers can function in multiple roles within an identity management relationship. These roles come with certain responsibilities – whether their role is Principal, Service Provider, Identity

Provider, Attribute Provider or Discovery Service. This paper reviews these roles and responsibilities in the context of privacy and security.

Security, Internet Protocols and Browsers:

- In developing its specifications, Liberty has evaluated the weaknesses of several well-known and frequently used Internet protocols and browsers. The Liberty Alliance has made every effort to provide secure standards, but since the standards are built on top of insecure protocols, there are some unavoidable potential vulnerabilities. This should not be misconstrued to suggest that the Liberty Specifications are insecure, but that implementations are dependent on all the underlying protocols, and thus care must be taken in implementation. The Liberty specifications neither increase nor eliminate these vulnerabilities.
- To that end, this best practices document provides definitions and information about security vulnerabilities and offers mitigating information to avoid the most common of them.

2. Introduction

The Liberty Alliance Project ("**Liberty Alliance**" or "**Liberty**") is an unincorporated, contract-based group of more than 150 companies and organizations from around the world. Liberty's objective is to create open, technical specifications ("**Liberty Specifications**") that (i) enable simplified sign-on through federated network identification on all current and emerging network access devices, and (ii) support and promote permissions-based attribute sharing to enable a user's ("**Principal's**") choice and control over the use and disclosure of such Principal's personal information. Liberty anticipates that these specifications will expedite the growth of e-commerce because they are designed to increase consumer convenience and confidence and to provide businesses with new business and cost-saving opportunities.

Liberty envisions that organizations will implement the Liberty Specifications in connection with their web-based offerings. Because privacy is important in these contexts, the Liberty Specifications include the necessary features and facilities to enable an implementing company to comply with its national privacy laws and regulations, or in the absence of law or regulation, best practices. Thus the Liberty Specifications will enable companies to adhere to information practices that comply with those laws and regulations.

The Liberty Alliance offers the guidance set forth in this document to implement the Liberty Specifications in an appropriately secure and privacy-friendly manner. Liberty also provides guidelines regarding privacy and security in a variety of documents, and intends to provide such guidance for future versions of the Liberty Specifications as well.[1]

This document first presents Liberty's perspective on privacy, followed by a discussion of certain general privacy laws and fair information practices. It then highlights certain "best practices" that will help those using the Liberty Specifications to ensure that privacy concerns are addressed. We offer some observations on security issues generally and Internet protocols and browsers more specifically. Finally, also attached for easy reference are links to books, papers, and other materials that discuss online security and privacy issues, as well as a broad sample of the variety of contemporary privacy paradigms that exist.

Companies that implement the Liberty Specifications are advised to consult with local counsel to ensure that the services they provide, based upon the Liberty Specifications, comply with applicable law. Please note that these best practices are intended to provide guidelines, not serve as an exhaustive resource. Furthermore, from a technical standpoint, these best practices are non-normative – they are not the rules defining the Liberty Specifications, but rather identify the privacy and security concerns that should be addressed when implementing Liberty Specifications. It is important to note that the Liberty Specifications are based upon existing Internet architecture and well-known protocols. The Liberty Specifications cannot and do not cure the well documented security challenges inherent in the

[1] Further guidelines on privacy and security can be found in the following Liberty specification documents: Liberty Alliance, "Liberty Security Bulletin October 11, 2002;" Thomas Wason, "Architectural Overview Version 1.0-2.0;" Paul Madsen, "Liberty Authentication Context Specification;" Gary Ellison, "ID-WSF Security Mechanisms;" Susan Landau, "ID-WSF Security and Privacy Overview;" John Kemp and Tom Wason, "ID-FF Bindings and Profiles;" Scott Cantor and John Kemp, "Liberty ID-FF Protocols and Schema Specification;" and John Linn, "Liberty Trust Models."

architecture of the Internet. Because companies from any part of the world, whether for-profit or not-for-profit, whether or not a member of the Liberty Alliance, may use the Liberty Specifications, the best practices identified in this document cannot and do not capture or address each potentially applicable legal privacy regime. Companies that implement the Liberty Specifications are advised to consult with local counsel to ensure that the services they provide, based upon the Liberty Specifications, comply with applicable privacy laws and regulations.

In addition, readers of this document should be aware that the Liberty Specifications are just that – specifications only. Given the global nature of e-commerce, the myriad of laws that apply to privacy, and the fact that the Liberty Alliance itself does not provide any services, the Liberty Alliance cannot and does not (i) advise as to what laws, regulations, or fair information practices are applicable to any given company, (ii) condition use of the Liberty Specifications on adoption of a particular set of fair information practices, (iii) monitor, audit or enforce compliance with applicable laws and regulations, nor (iv) have any liability with respect to an implementing company's use of the Liberty Specifications. The implementing companies remain responsible for monitoring implementation and, as is the case today, remain answerable to local enforcement authorities for non-compliance with applicable laws.

Similarly, implementing companies should monitor the ever-changing status of security challenges, and should take these into account when designing their implementations. To aid in this effort, we present in this document some of the well-known Internet and protocol security vulnerabilities that should be taken into account when implementing the Liberty Specifications.

3. Liberty Alliance Perspective on Privacy

The Liberty Alliance considers privacy and security of a Principal's personal information to be extremely important. Privacy, security and consumer considerations drive many decisions the Liberty Alliance made about the world of online commerce that the Liberty Specifications enable. In particular, the Liberty Alliance made the following fundamental decisions regarding the Liberty Specifications:

- To use a de-centralized architecture, where it is not necessary to have data stored with a single entity;
- To use a federated architecture, where parties are free to link networks as business judgment dictates;
- To support and promote permissions-based attribute sharing to enable consumer choice and control over the use and disclosure of his or her personal information;
- To provide open specifications that are not centrally administered;
- To provide interoperable specifications that can be used on a wide variety of network access devices;
- To leverage existing systems, standards, and protocols where they work well;
- To enable companies to transmit information using the specifications with the best available security;
- To include in the specifications, tools that enable companies to respond to consumer interests regarding privacy and security and to compete on that basis.

Our perspective on privacy is necessarily informed by two key characteristics of our work:

1. The Liberty Alliance is a group of individual companies writing open technical specifications and, except for the specifications, the Liberty Alliance does not provide products or services directly to the public; and
2. The Liberty Alliance is composed of individual member companies from around the globe and from myriad sectors of the economy. Many of the Liberty Alliance members serve consumers directly and others create the infrastructure products that are used by companies to serve consumers directly and by companies to run their internal operations. Some companies serve global markets, and others serve regional markets in Europe, Japan, Korea, and the United States.

Given the goals of facilitating e-commerce and providing a mechanism that permits compliance with local law and appropriate security, Liberty believes that the responsible approach is to create flexible, interoperable specifications that can be implemented around the globe in a variety of different ways to satisfy applicable privacy and security concerns and related laws. Thus, the Liberty Specifications provide tools that can be used by implementing companies to address privacy and security concerns. As noted earlier, the implementing companies are solely responsible for

deploying the Liberty Specifications in a secure manner that complies with applicable privacy laws and fair information practices.

Because the Liberty Specifications represent a novel approach to account linking and data exchange, we believe it is important to identify implementation best practices to accompany the specifications. The best practices below simply explain what fair information principles Liberty-enabled providers should address, depending upon which role(s) they perform. This document also explains the tools contained in the specifications that can be used to respond to such considerations, including elements of the Liberty Specifications that enable implementers to more effectively utilize various rights expression languages to communicate information about usage directives that may be associated with a given attribute.

Finally, it is important to note that the Liberty Specifications are built on a framework that presumes data exchange of personal attributes will occur in the context of permissioning. While, as noted, the Liberty Alliance has no role in providing services, many of the architectural decisions made in creating the Liberty Specifications were made on the presumption that those providing services based on the Liberty Specifications would be engaging in permissions-based attribute sharing.

These architectural considerations, as well as specific features of the Liberty Specifications reflect the fact that the Liberty Alliance is both very conscious of the importance of privacy, security and other public policy considerations, and cognizant of the fact that the Liberty Alliance is, fundamentally, an open specifications body that cannot – and should not – enforce particular implementations on parties using the Liberty Specifications.

4. Privacy laws

There are a variety of privacy protection laws throughout the world, each with its unique set of requirements and obligations. The following discussion highlights some, but by no means all, of these laws, and the differences between those highlighted.[2]

Some of the privacy laws that have been enacted include, among others:

- European Union Directive on Data Protection of individuals with regard to the processing of personal data and the free movement of such data.[3]
- European Union Directive concerning the processing of personal data and the protection of privacy in the electronic communications sector.[4]
- In Canada – The Personal Information Protection and Electronic Documents Act.[5]
- In the United States – The Children's Online Privacy Protection Act, The Graham-Leach-Bliley Act, The Health Insurance Portability and Accountability Act, and more generally, the Federal Trade Commission has challenged online privacy polices under Section 5 of the Federal Trade Commission Act.[6]
- In Other Territories – According to Privacy International's Privacy and Human Rights: An International Survey of Privacy Laws and Developments, 2003, there are several other territories that have enacted or enforced some form of privacy laws.[7]

The EU Data Protection Directive (95/46/EC) aims to protect the privacy of EU citizens and harmonize differing laws and regulations in the Member States. The EU Data Protection Directive covers any information relating to an

[2] The summaries of law provided below are for informational purposes only, and are not intended to be legal advice.

[3] EU Data Protection Directive 95/46/EC.

[4] EU Directive on Privacy and Electronic Communications 2002/58/EC.

[5] Canadian Privacy Act. S.C. 2000.

[6] See 15 U.S.C.A. § 6501 (2000); 15 U.S.C.A. § 6801 et. seq. (1999); 42 U.S.C.A. §§ 1320(d) et. seq. (1996); 15 U.S.C.A. § 45(a) (2000).

[7] Privacy International, "Privacy and Human Rights: An International Survey of Privacy Laws and Developments, 2002."

individual, even if collected purely in a business context, such as contact details of business clients, and not just consumers. Data about a company's employees is also covered.

The Directive imposes duties on data "controllers," those who determine the purpose and means of processing data. In addition to imposing registration requirements with data commissioners in Member States, the Directive requires that data be (i) processed fairly, (ii) collected for specified, explicit and legitimate purposes and not used in ways incompatible with those purposes, (iii) collected only to the extent that it is adequate and relevant, and not excessive in relation to the purposes for which it was collected, (iv) accurate and kept up to date, (v) kept no longer than necessary for the purposes for which it was collected, and (vi) not transferred to third-party countries that do not ensure an adequate level of protection for the data. Data may generally be processed either by unambiguous consent or where necessary to perform a contract. However, certain data is considered sensitive requiring explicit consent before processing. In addition, data subjects are to be given notice of the data controller, purposes of processing their data, recipients of the data, and the right to access and correct the data. This notice to the data subject is to be provided even when the data has been obtained not from the data subject but from a third party.

The EU Data Protection Directive for electronic communications (2002/58/EC) complements the aforementioned Directive for the electronic communications sector. It includes inter alia specific provisions regarding the confidentiality of communications, the handling of traffic and location data and the requirements and restrictions applying to the use of cookies as well as to unsolicited electronic communications. Member States are required to implement the Directive by 31 October 2003.

The Canadian Privacy Act sets out ground rules for how private sector organizations may collect, use or disclose personal information in the course of commercial activities. The Act is centered around ten (10) underlying principles regarding (i) accountability, (ii) identifying purpose, (iii) consent, (iv) limiting collection, (v) limiting use, disclosure, and retention, (vi) accuracy, (vii) safeguards, (viii) openness, (ix) individual access, and (x) challenging compliance.

United States laws, on the other hand, take a more vertical approach with express federal privacy related statutes for specific sectors. For example, the Graham-Leach Bliley-Act ("**GLB Act**") governs the use of personal information that is given to and managed by financial service providers. The Health Insurance Portability and Accountability Act ("**HIPAA**") mandates standards for the use of protected health care data. The Children's Online Privacy Protection Act ("**COPPA**") governs the use and collection of personal information about children under age 13. Each of these laws requires some form of notice and consent before disclosure of the personal information at issue, and requires that a party use reasonable safeguards to maintain the confidentiality of such personal information. Under the GLB Act, for example, consent to share personal information with unaffiliated third parties for marketing purposes via an "opt-out" procedure rather than express or an "opt-in" consent, may be acceptable. In other cases, the Federal Trade Commission has exercised its authority under Section 5 of the Federal Trade Commission Act to challenge certain privacy practices as "unfair and deceptive."[8]

Given the breadth and variety of privacy laws that may be applicable depending upon the jurisdiction in which a company does business, the Liberty Alliance strongly recommends that any entity implementing the Liberty Specifications consult with local counsel to determine which laws are applicable to the company's business and how best to comply with those laws.

5. Fair Information Principles

General. In addition to existing privacy laws, several organizations have set forth fair information practices governing the use and disclosure of personal information. These organizations include, among others, the Online Privacy Alliance ("**OPA**"), the Organization for Economic Co-operation and Development ("**OECD**"), the Center for Democracy and Technology ("**CDT**"), the Network Advertising Initiative ("**NAI**"), Health Internet Ethics ("**Hi-Ethics**"), and the Global Business Dialogue on Electronic Commerce ("**GBDe**").[9]

[8] For an overview of U.S. federal and state privacy laws, see BBB Online, Inc. and the Council of Better Business Bureaus, Inc., "A Review of Federal and State Privacy Laws."

[9] Links to several of these fair information practices or privacy guidelines are noted in the References section of this document.

There are no universal standards among these organizations as to what fair information practices entail. The differences seen in these fair information practices are partially attributable to geography, and partially attributable to the sector to which the fair information practices apply.

OPA Guidelines. OPA is a U.S.-based organization which provides a general framework in which any U.S. company can operate, calling for customization and enhancement as appropriate to a company's business or industry sector. These guidelines generally provide as follows:

- ***Adoption and Implementation of a Privacy Policy*** – An organization should adopt and implement a policy for protecting the privacy of individually identifiable information.
- ***Notice and Disclosure*** – The privacy policy should be clear, easy to find, and available at or prior to the time individually identifiable information is collected. It should state "what information is being collected; the use of that information; possible third-party distribution of that information; the choices available to an individual regarding collection, use and distribution of the collected information; a statement of the organization's commitment to data security; and what steps the organization takes to ensure data quality and access. It should also disclose the consequences, if any, of an individual's refusal to provide information. The policy should also include a clear statement of what accountability mechanism the organization uses, including how to contact the organization."
- ***Choice/Consent*** – Individuals must be given the opportunity to exercise choice regarding how individually identifiable information collected from them online may be used when such use is unrelated to the purpose for which the information was collected or where there is third-party distribution of such data unrelated to the purpose for which it is collected. At a minimum, individuals should be given the opportunity to opt out of such use or third-party distribution.
- ***Data Security*** – Organizations creating, maintaining, using or disseminating individually identifiable information should take appropriate measures to assure its reliability and should take reasonable precautions to protect it from loss, misuse or alteration. Organizations should take reasonable steps to assure that third parties to which they transfer such information are aware of these security practices, and that the third parties also take reasonable precautions to protect any transferred information.
- ***Data Quality and Access*** – Organizations creating, maintaining, using or disseminating individually identifiable information should take reasonable steps to assure that the data are accurate, complete and timely for the purposes for which they are to be used. Organizations should establish appropriate processes or mechanisms so that inaccuracies in material individually identifiable information, such as account or contact information, may be corrected. These processes and mechanisms should be simple and easy to use, and provide assurance that inaccuracies have been corrected. Other procedures to assure data quality may include use of reliable sources and collection methods, reasonable and appropriate consumer access and correction, and protections against accidental or unauthorized alteration.

OECD Guidelines. The OECD is an international organization focusing on global economic cooperation and development. OECD guidelines on the protection of privacy and trans-border flows of personal data are close to the European approach to privacy. Unlike the United States, Europe has more comprehensive privacy statutes and vests significant authority in its regulatory bodies to enforce privacy legislation. OECD's guidelines set forth the following eight principles:

- ***Collection Limitation*** – There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject.
- ***Data Quality*** – Personal data should be relevant to the purposes for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date.
- ***Purpose Specification*** – The purposes for which personal data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfillment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of purpose.
- ***Use Limitation*** – Personal data should not be disclosed, made available or otherwise used for purposes other than those specified in accordance with Article 9 (the purpose specification principle) except: (a) with the consent of the data subject; or (b) by the authority of law.

- ***Security Safeguards*** – Personal data should be protected by reasonable security safeguards against such risks as loss or unauthorized access, destruction, use, modification or disclosure of data.

- ***Openness*** – There should be a general policy of openness about developments, practices and policies with respect to personal data. Means should be readily available of establishing the existence and nature of personal data, and the main purposes of their use, as well as the identity and usual residence of the data controller.

- ***Individual Participation*** – An individual should have the right: (a) to obtain from a data controller, or otherwise, confirmation of whether or not the data controller has data relating to him; (b) to have communicated to him, data relating to him within a reasonable time; at a charge, if any, that is not excessive; in a reasonable manner; and in a form that is readily intelligible to him; (c) to be given reasons if a request made under subparagraphs (a) and (b) is denied, and to be able to challenge such denial; and (d) to challenge data relating to him and, if the challenge is successful, to have the data erased, rectified, completed or amended.

- ***Accountability*** – A data controller should be accountable for complying with measures which give effect to the principles stated above.

6. Liberty Alliance Privacy Recommendations

As evident from the preceding sections, there is a wide range of fair information practices that have been promoted around the world. In an effort to promote "best practices," Liberty recommends that an implementing company comply with all relevant laws. In the absence of laws, an implementing company should follow the most appropriate fair information practices applicable to the jurisdiction and industry sector in which the company intends to do business or offer products or services. Where applicable, an entity should not request or provide more information than is necessary for the interaction.

In addition, the Liberty Alliance offers the following baseline set of fair information practices as guidelines that companies, whether in the role of Service Provider, Identity Provider, Attribute Provider, Discovery Service or otherwise, should consider adopting when implementing Liberty Specifications. These recommended fair information practices are based on principles of notice, choice and control, access, security, quality, relevance, timeliness and accountability. Each of these practices, and its appropriateness in the context of Liberty Services, is briefly described below.

- ***Notice.*** Consumer facing Liberty-Enabled Providers should provide to the Principal clear notice of who is collecting the information, what information they collect, how they collect it (e.g., directly or through non-obvious means, such as cookies), how they provide choice, access, security, quality, relevance and timeliness to Principals, whether they disclose the information collected to other entities, and whether other entities are collecting information through them. Providing notice is particularly important for Service Providers who may seek additional information beyond what is provided through other Liberty-Enabled Providers.

- ***Choice.*** Consumer facing Liberty-Enabled Providers should offer Principals choices, to the extent appropriate given the circumstances, regarding what personally identifiable information is collected and how the personally identifiable information is used beyond the use for which the information was provided. In addition, consumer facing Liberty-Enabled Providers should allow Principals to review, verify, or update consents previously given or denied. The Liberty Specifications provide for both access permissions to allow a Principal to specify whether and under what circumstances a Service Provider can obtain given attributes, as well as an "envelope" for the discovery of or negotiation of usage directives as part of profile sharing. Both aspects of the privacy capabilities established by the Liberty Specifications should be fully implemented in a responsible manner and be easy for the Principal to configure. In particular, Liberty-Enabled Providers should provide for "usage directives" for data through either contractual arrangements, or through the use of Rights Expression Languages, as well as implementing the access authorization elements contained in the Liberty Specifications that permit the Principal to make certain choices regarding collection and use of personally identifiable information.

- ***Principal Access to Personally Identifiable Information (PII).*** Consumer facing Liberty-Enabled Providers that maintain PII should offer, consistent with and as required by relevant law, a Principal reasonable access to view the non-proprietary PII that it collects from the Principal or maintains about the

Principal. Access should not be construed to require access to proprietary data, public record data, or aggregate data.

- ***Quality.*** Consumer facing Liberty-Enabled Providers that collect and maintain personally identifiable information should permit Principals a reasonable opportunity to provide corrections to the personally identifiable information that is stored by such entities.
- ***Relevance.*** Liberty-Enabled Providers should use PII for the purpose for which it was collected, or the purposes about which the Principal has consented.
- ***Timeliness.*** Liberty-Enabled Providers should retain PII only so long as is necessary or requested and consistent with a retention policy accepted by the Principal.
- ***Complaint Resolution.*** Liberty-Enabled Providers should offer a complaint resolution mechanism for Principals who believe their PII has been mishandled.
- ***Security.*** Liberty-Enabled Providers should take reasonable steps to protect and provide an adequate level of security for PII.

Implementing companies should be aware that the Liberty Specifications provide tools the implementing company can use to help it comply with fair information practices, regardless of which protocol is adopted. These tools are discussed in more detail below. However, implementing companies should also be aware that any of these fair information practices (as well as any recommendations set forth in this document) are only guidelines, and may or may not satisfy or be consistent with the privacy laws, rules, and regulations applicable to the implementing company. Therefore, Liberty strongly recommends that any implementing company consult with local counsel to determine which laws are applicable to the company's business and how best to comply with those laws.

In order to address various privacy concerns and implement fair information practices using the Liberty Specifications, it is important for an implementing company to understand how certain schemas and protocols in the specifications operate and be aware of certain tools contained in the specifications that can be used to respond to such considerations.

Consumer choice and permission are central to Liberty's vision. The framework of the Liberty Specifications is built upon the presumption that PII will be shared ("attribute sharing") in the context of permissioning, i.e., upon the consent of the Principal and in accordance with the usages expressed by the Principal. Such attribute sharing should be predicated upon not only a prior agreement between the Liberty-Enabled Providers, but also upon providing notice to the Principal and obtaining the Principal's consent. The Liberty Specifications allow for recording both the notice and consent in an auditable fashion. Liberty recognizes that depending upon the particular implementation, for example in financial services transactions, it may be important to both the Principal and the Liberty-Enabled Provider to have increased certainty regarding their transaction. Such certainty may be achieved through the use of auditable records of notice and consent. In addition, Liberty-enabled providers should take reasonable measures to prevent unauthorized acquisition of a principal's personal information (e.g. by harvesting).

Within this framework, the Liberty Specifications identify various roles that comprise the federated identity infrastructure. Each of these roles has certain responsibilities in relation to protecting the privacy and security of a Principal's personally identifiable information. Liberty-Enabled Providers may function in multiple provider roles. These roles and their respective responsibilities include, among others:

- ***Principal*** – A Principal is an entity that can acquire a federated identity that is capable of making decisions and can be authenticated and vouched for by an Identity Provider. In a business-to-consumer (B2C) context, the Principal is the consumer. In other contexts, the Principal could be an individual, a corporation, or another legal entity. The fair information best practices set forth in this document are aimed to protect the confidentiality of the Principal's personally identifiable information. Principals should be vigilant when they provide their credentials (e.g., passwords, secure tokens, etc., entered over secure channels) or attributes over the web to protect themselves from being spoofed or otherwise providing such information to an unintended party. In addition, Principals should use care when establishing or modifying credentials. Also, Principals should become familiar with an entity's posted data practices before providing personally identifiable data to such entity.
- ***Service Provider*** – A Service Provider is an entity that provides services to Principals. Liberty envisions that the Service Provider will, upon request from a Principal, request that the Identity Provider (which may be itself or another party) authenticate the Principal. After the Principal has been authenticated, the Service Provider may request that certain attributes regarding the Principal be

provided to it in order to provide the requested services to the Principal. Service Providers should inform Principals of their data practices, provide the Principal with certain choices regarding secondary uses of the Principal's personally identifiable information, maintain the security of a Principal's personally identifiable information within their control, and not use or share such information except in accordance with the Service Provider's privacy policy and/or the consent or usage directives of the Principal. The Service Provider should at all times ensure that its data practices conform with applicable local law and practice.

- ***Identity Provider*** – An Identity Provider is an entity that creates, maintains, and manages identity information for Principals. It authenticates and vouches for the Principal to other Service Providers within an Authentication Domain. The Identity Provider should also safeguard the Principal's identity credentials, and have some mechanisms in place to require the Authentication Domain to use the credentials in a proper manner.

- ***Attribute Provider*** – An Attribute Provider is an entity that provides attributes to a requester (i.e., a Service Provider) in accordance with its own policies and a Principal's permissions. Attribute Providers store and negotiate access control information defining the circumstances under which a Service Provider will be granted access to a given attribute(s). Attribute Providers store and negotiate usage directives that specify the manner in which attributes can be used, stored, and disclosed. An Attribute Provider has at least the same responsibilities as Service Providers with respect to clear notice (including notice to the Principal regarding what are the default usage directives and how the Principal can change such usage directives), choice, security, and responsible use and sharing of a Principal's data.

- ***Discovery Service*** – A Discovery Service is an entity (usually an Identity Provider) that has the ability to direct attribute requesters to the relevant Attribute Provider who provides the requested classes of attributes for the specified Principal. The Discovery Service should register only those Attribute Providers in accordance with the consent or usage directives of the Principal. The Discovery Service should permit the Principal to see which Attribute Providers have been registered on the Principal's behalf. An attribute requester can locate the Attribute Providers for a given Principal, even though the attribute requester and Attribute Providers do not have a common name for the Principal.

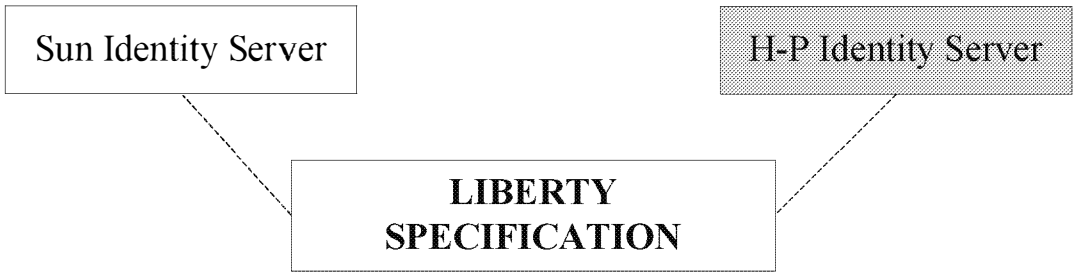
The following are two hypothetical examples of implementation of the Liberty Specifications by various companies. In Example 1, we start with the actual equipment manufacturers who could build Internet infrastructure software utilizing the specifications. Example 1 goes through the steps to an end user experience. Example 2 is a one page summary illustration of the process undertaken and explained in Example 1. Where Example 1 starts with software manufacturer, Example 2 shows the same process, in summary form, starting with the user.

393 **Hypothetical Example 1**

394 *1. Technology Implementation*

395 a. Sun develops an identity server (infrastructure software) using the Project Liberty Alliance
396 specifications (Liberty Specification) in order to allow the Sun identity server to interoperate with other
397 technology products that implement the Liberty Specification.

398 b. H-P develops an identity server (infrastructure software) using the Liberty Specification in order to
399 allow the H-P identity server to interoperate with other technology products that implement the Liberty
400 Specification.



401

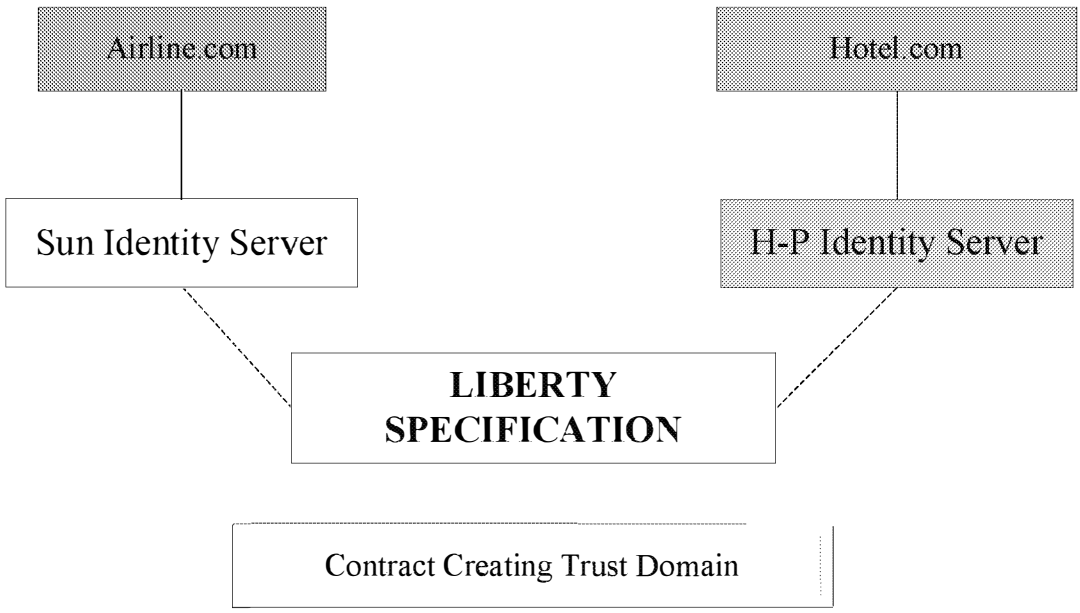
402

403

2. *Services Deployment*

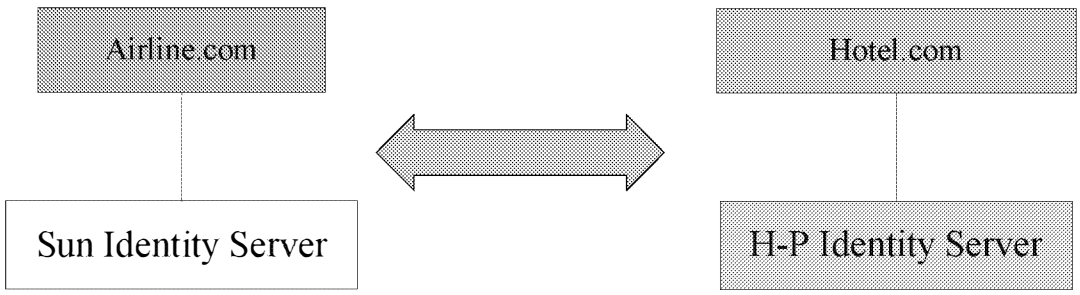
a. Airline employs a Sun Identity Server to authenticate users to its website, maintain frequent flyer information, and provide linkage to its reservation system.

b. Hotel uses an HP Identity Server to authenticate users to its website and facilitate online reservations.



3. *Trust Domain*

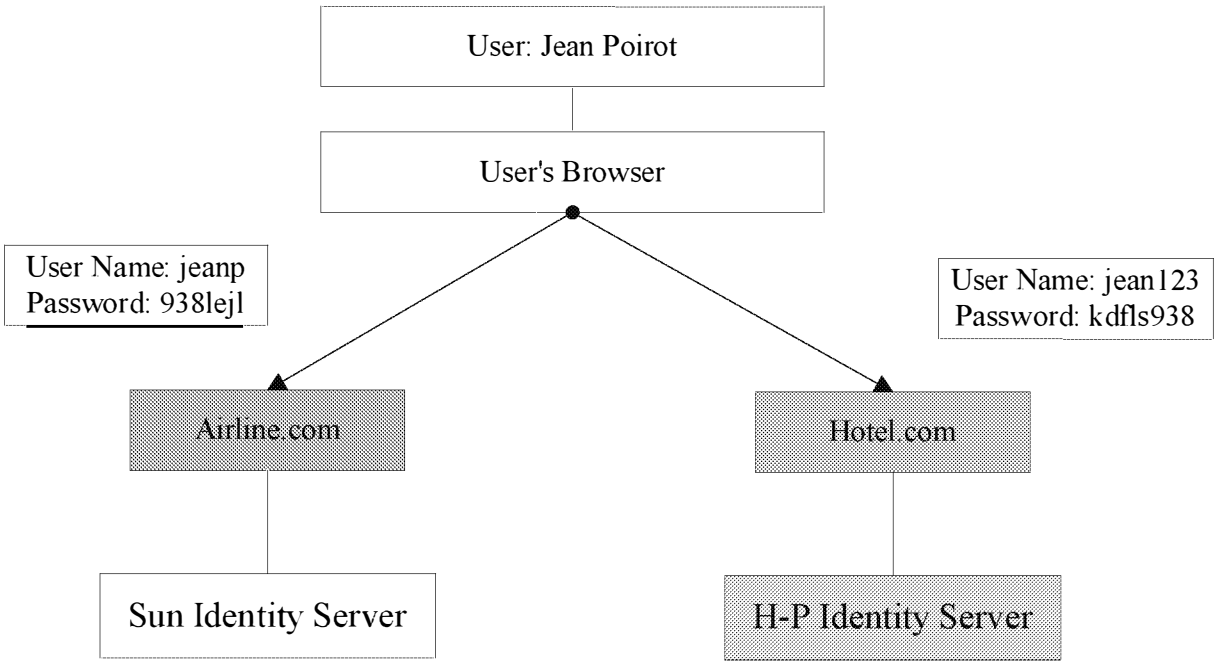
a. Airline and Hotel enter into a contract, which requires, among other things, that Hotel will accept Airline's authentication of a customer that is a customer of both Airline and Hotel (Airline is an Identity Provider and Hotel is a Service Provider.)



4. *Account Establishment*

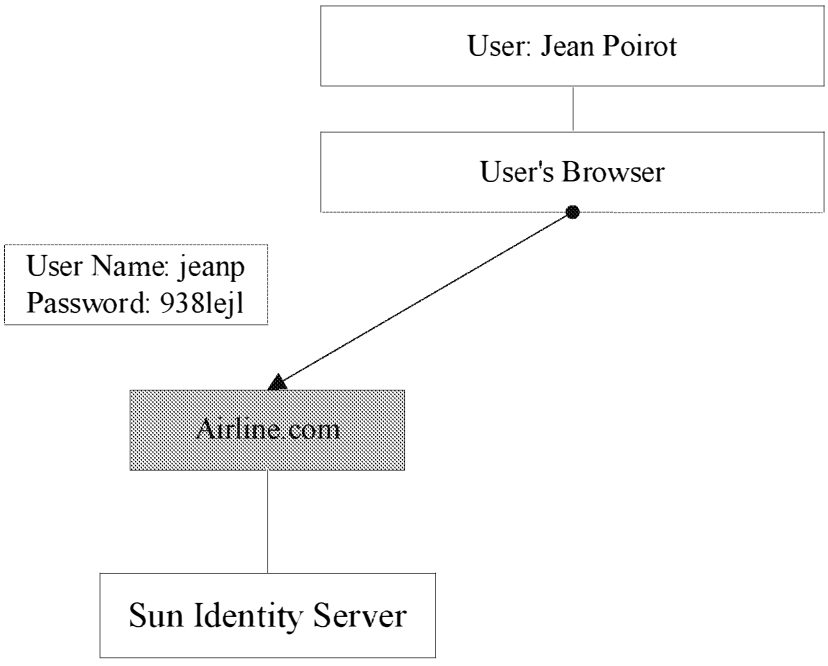
a. User establishes an account with Airline.com with User ID JeanP and a password.

b. User establishes an account at Hotel.com with user name Jean123 and a password.

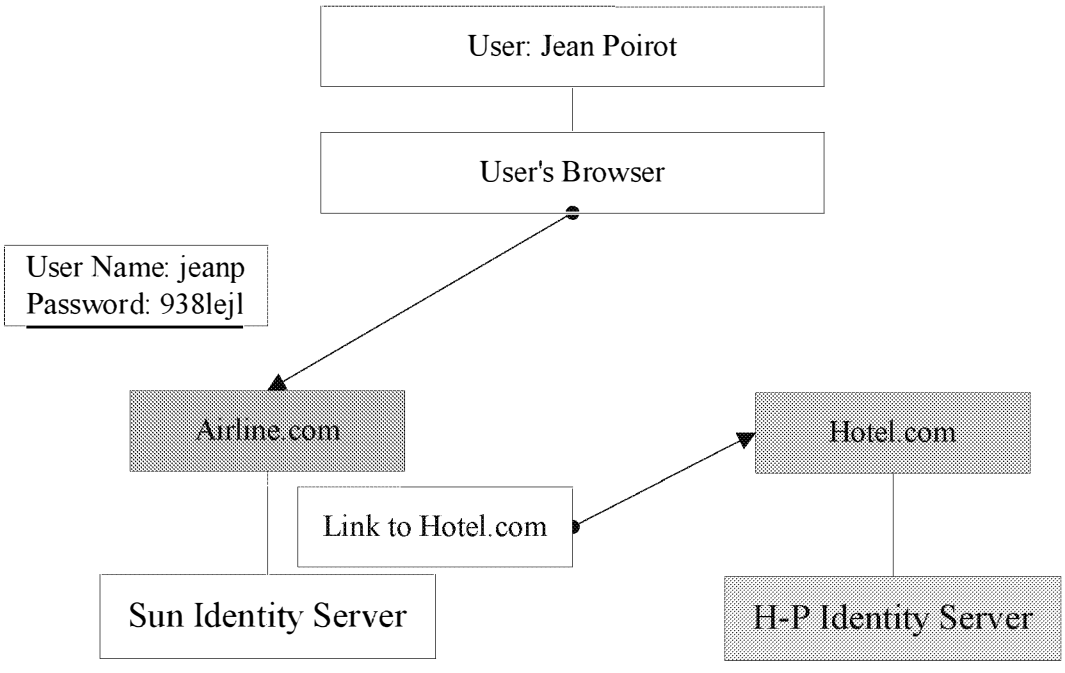


5. *Linking Accounts*

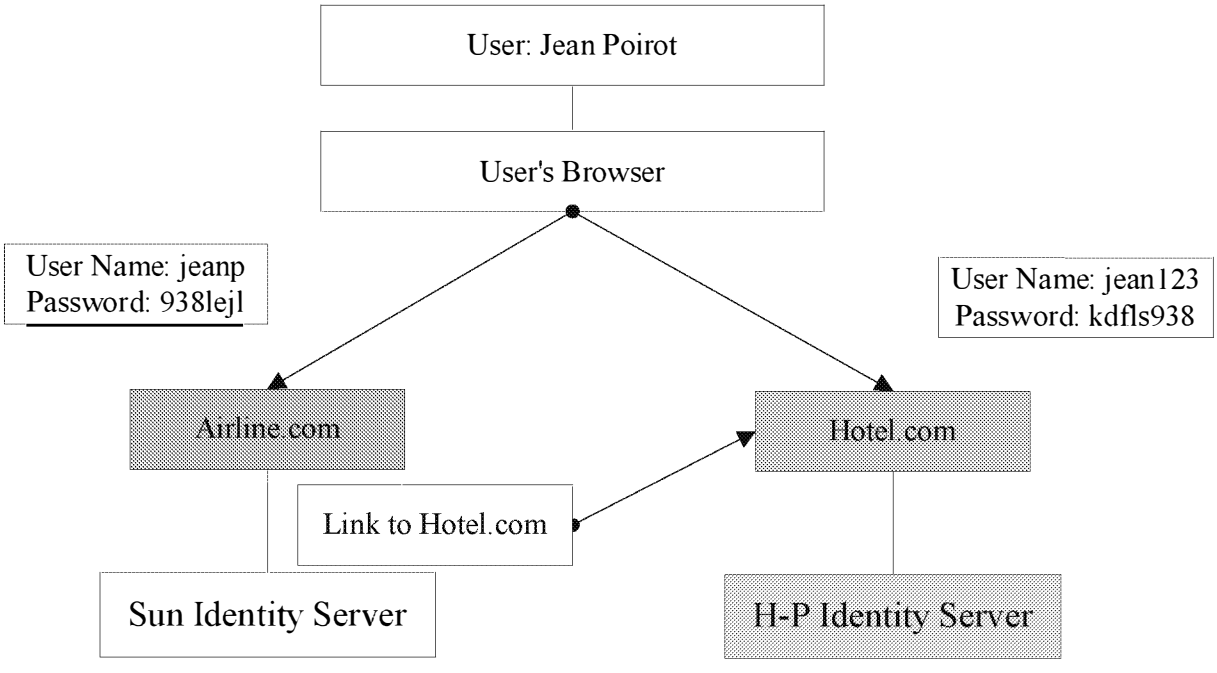
a. Jean logs onto the Airline site with user name and password



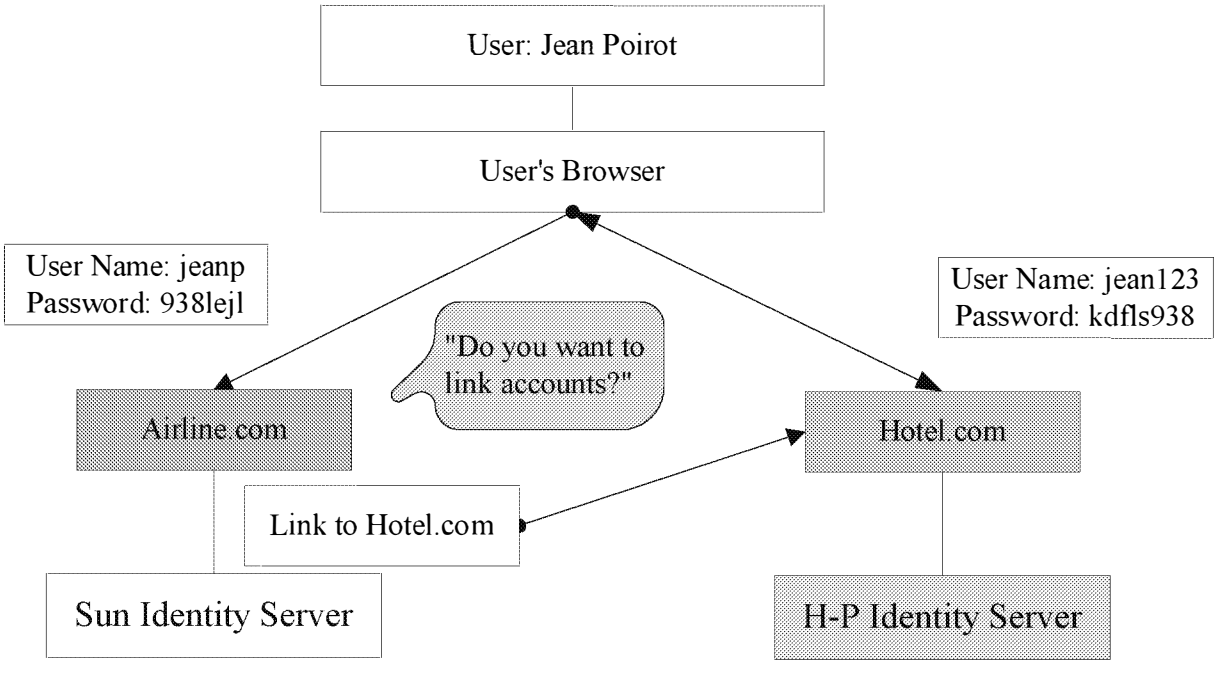
b. Airline has a link that JeanP can use to go to the Hotel website.



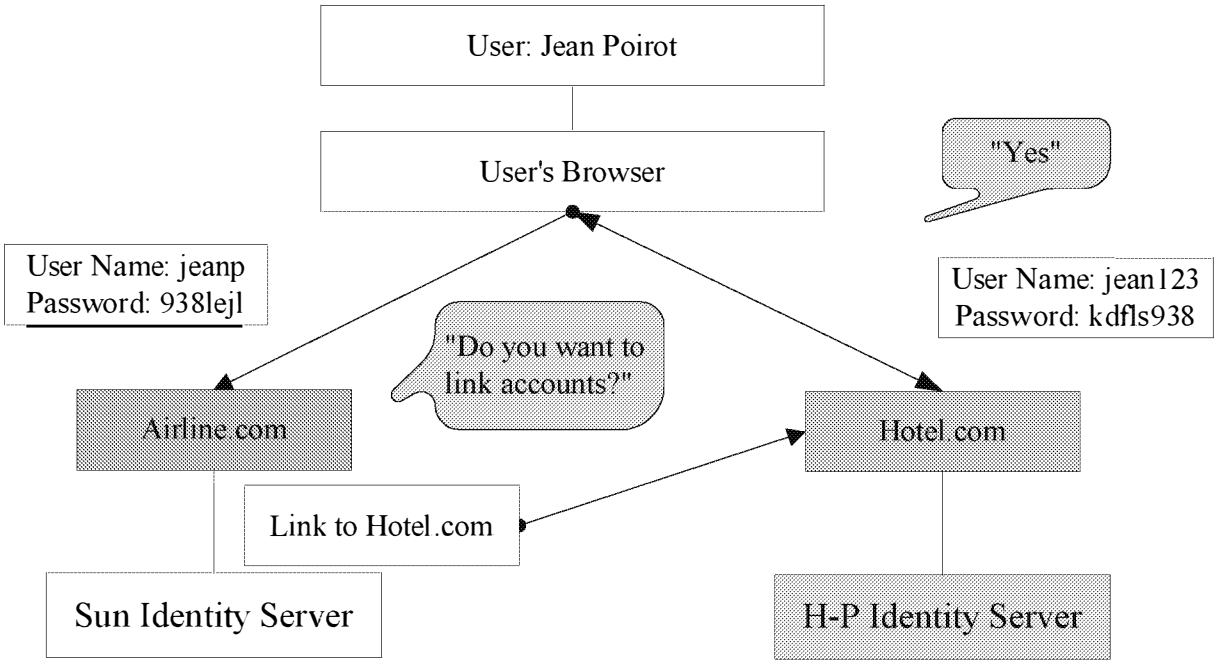
c. JeanP clicks on that link and goes to the Hotel website and logs in with username Jean123 and password.



d. Hotel would know that Jean123 came from the Airline site and asks if Jean123 would like to link his account at Airline with his account at Hotel, enabling single sign-on between the two accounts.



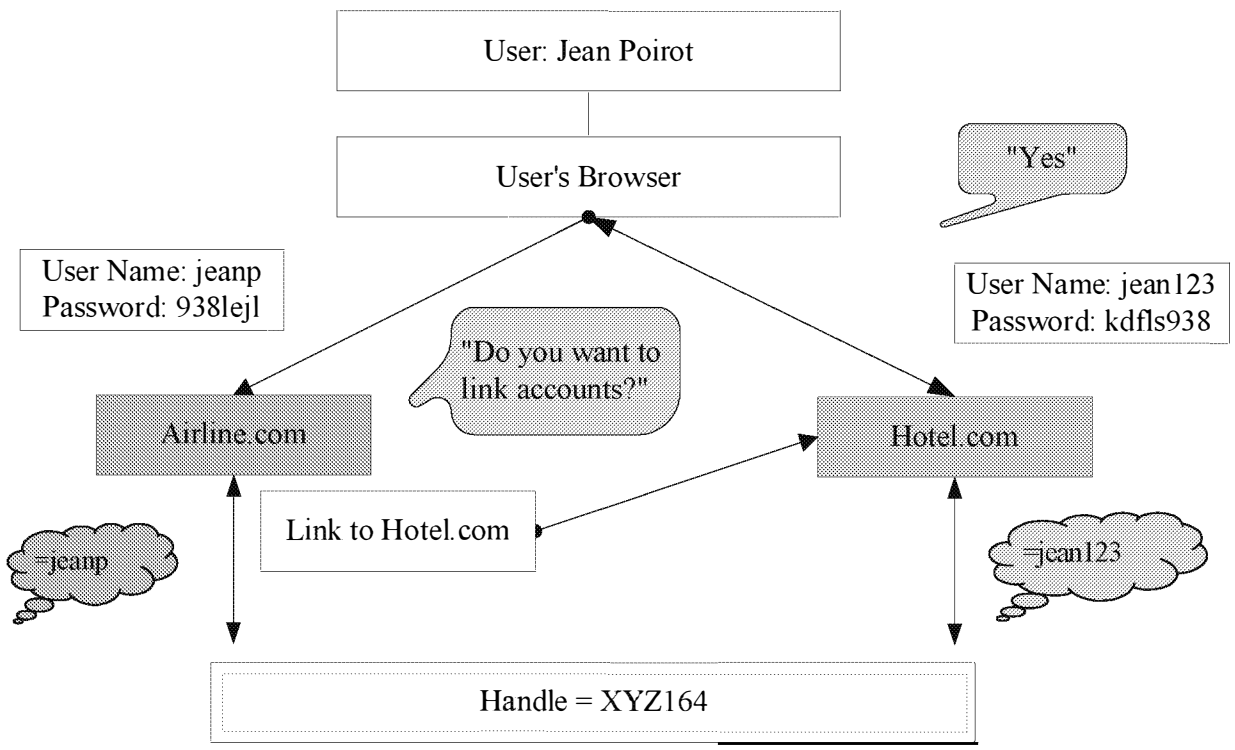
439 e. Jean123 indicates his consent by clicking "yes".



440

441

442 f. Airline and Hotel agree on a random set of characters (handle) by which each will recognize user as
443 their customer (e.g., Airline: XYZ164 = authenticated JeanP; Hotel: XYZ164 = authenticated Jean123)



444
445

446

447 *6. Simplified sign-on (SSO)*

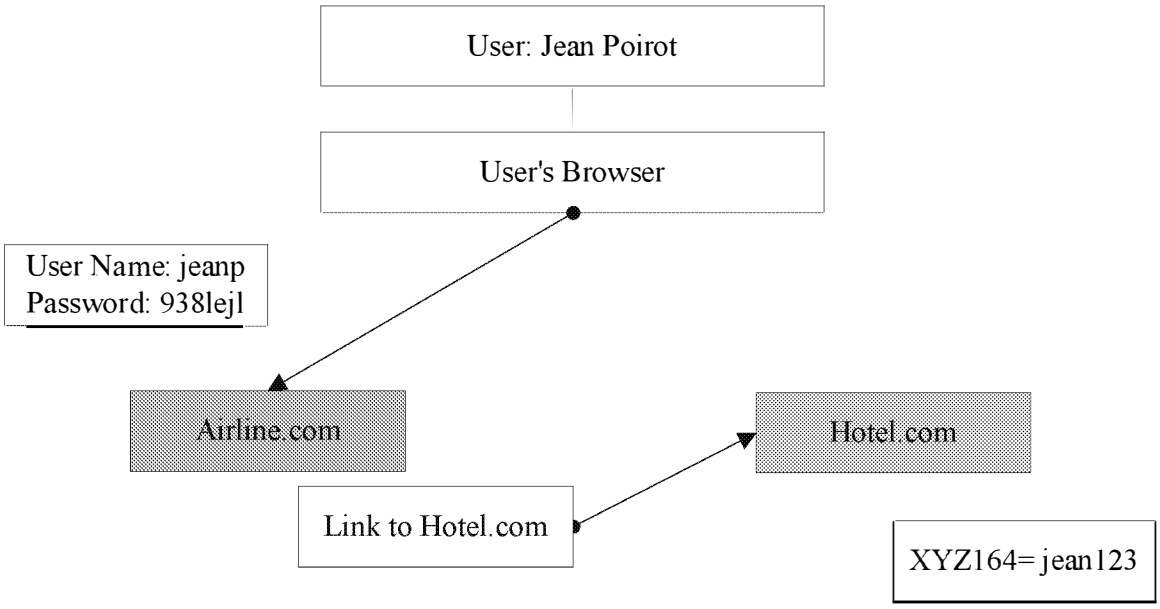
448 a. User goes to Airline and logs in as JeanP and password.

449 b. User clicks on link to Hotel website.

450 c. Airline sends opaque handle to Hotel site with username.

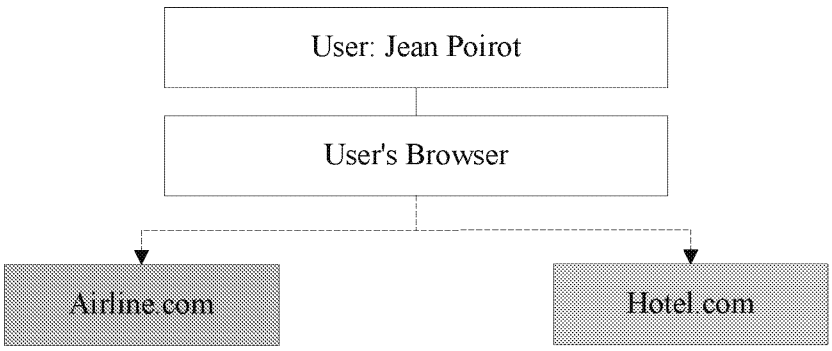
451 d. Hotel receives opaque handle and recognizes user as authenticated Jean123.

452



453

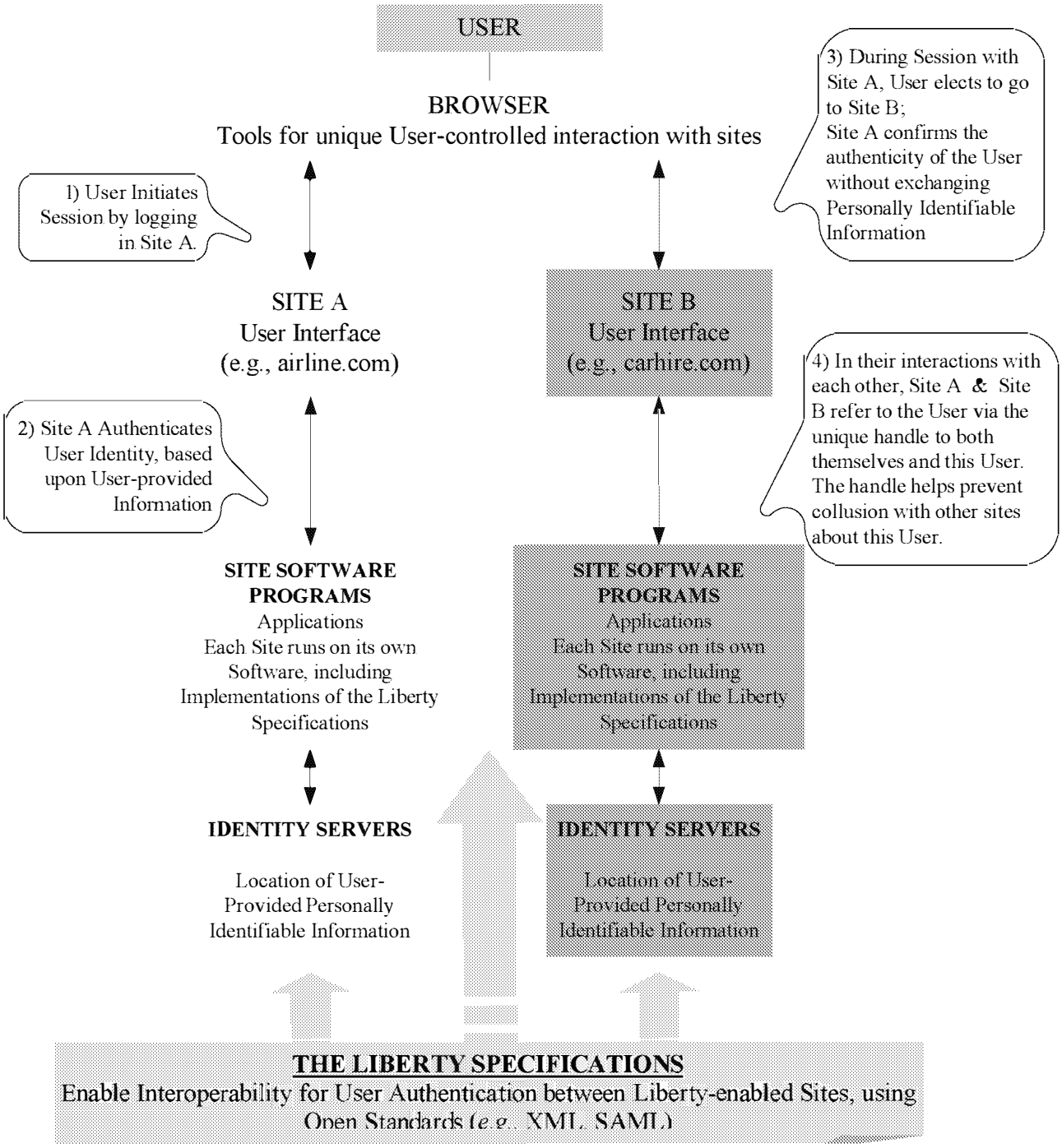
454 e. When Jean logs out of either site, he is automatically logged out of both sites.



455
456

457

457 The following diagram is a summary hypothetical example of deployment by various companies:



458
459

460

461 In either example, if starting at an Identity Provider, the Principal makes an initial authentication by presenting a user
462 name (real or pseudonymous) and a corresponding password. Next, the Principal establishes a local account with
463 various Service Providers. The final step is for the Principal to link accounts. Account linking will likely be a two-step
464 process. First the Principal consents to account linking and then provides specific consent for each Service Provider.
465 After these steps, the Principal will be able to benefit from SSO and surf the Internet in a Trust Domain using the same
466 name and password given initially to the Identity Provider.

Alternatively, a Principal may wish to connect to a Service Provider without first going to an Identity Provider. In this case the Service Provider will typically provide the Principal with a list of all Identity Providers with whom the Service Provider has formed a Trust Domain, and the Principal will be offered the opportunity to click on its preferred Identity Provider in order to authenticate. The Service Provider will redirect the Principal's browser to the chosen Identity Provider who will recognize the Principal by name and password (which the Principal will have to disclose). The Principal is then authenticated for all Providers within that Trust Domain for this session of Internet use.

At the end of the session, the Principal will benefit from an automatic Single-Log-Out performed by the system, i.e., by logging out of any site, the Principal can be automatically logged out of all of the other sites visited during that particular session of Internet use.

Of course, it is also possible for a Principal to visit a single Service Provider and log into that Service Provider using its user name and password recognized by that Service Provider, but it will not be able to take advantage of SSO until it authenticates at an Identity Provider.

In the preceding illustration, Jean now has a federated identity between Airline and Hotel. Assume that Jean has also federated his identity at Bank with his Airline and Hotel identities. Jean's identity at Airline includes Jean's name, address, and frequent flier information. Jean's account at Bank includes an electronic wallet where Jean stores his credit card information. Bank has previously notified Airline that it has credit card information related to Jean. Jean logs in at the Airline website, visits the Hotel website, and desires to make a reservation for a hotel room. In order to take a reservation, Hotel needs information regarding Jean's name, address, and credit card information. Hotel requests this information from Airline. After checkings Jean's preferences regarding data, Airline provides the name and address information to Hotel, and directs Hotel to obtain credit card information from Bank. After checking Jean's preferences regarding data disclosure, Bank provides credit card information to Hotel. All of these interactions between Hotel, Airline, and Bank are done "behind the scenes." After Jean makes his request to make a reservation, his name, address, and credit card information are automatically included in the form presented to Jean at Hotel's website to make his reservation.

In this example, Airline acts not only as an Identity Provider (authenticating Jean's identity for Hotel), but also as an Attribute Provider (providing name and address information to Hotel) and as a Discovery Service (letting Hotel know that Jean's credit card information may be obtained from Bank). Bank is also an Attribute Provider because it provides credit card information to Hotel.

The roles of each of the Liberty-Enabled providers should be viewed within a privacy policy framework in which fair information practices are implemented. The framework should address both whether an attribute requester may obtain access to certain classes of a Principal's attributes, and if yes, what fine-grained methods are allowed, and what data is returned to the attribute requester.

In order to enable implementers to set up this framework, the Liberty Specifications include a number of tools designed to (i) increase a Principal's choice and control with respect to (a) the federation of his identity within an Authentication Domain and (b) use and disclosure of personally identifiable information, and (ii) facilitate certain interactions among Identity Providers, Service Providers, and Attribute Providers without disclosing a Principal's identity. These tools include:

- ***Access Controls*** – The Liberty Specifications enable Liberty Providers to make access control decisions on behalf of the Principal. Liberty-Enabled Providers should provide a mechanism by which the Principal can specify his or her authorization policy. Thus, for example, a Principal might not allow his or her home address to go to a newspaper website, but might allow it to be sent to an online store as part of a sales transaction.

- ***Usage Directives*** – The Liberty Specifications describe a container that may be used to list or point to usage directives regarding either the intended use of a requested attribute (from the requester), or the allowed usage of a requested attribute (from the attribute owner/holder). The Attribute Provider and the Service Provider may negotiate acceptable usage directives. The Attribute Provider can provide the Service Provider with a list of acceptable usage directives when the intended usage requested by the Service Provider doesn't match the Principal's usage directives.

- ***Opaque Handles*** – The Liberty Specifications support opaque handles, the assignment of an arbitrary sequence of characters by the Identity Provider or Service Provider to identify a Principal. The opaque handle has meaning only in the context of the relationship between the Identity Provider and the Service Provider. Thus a Principal's identity and actions are harder to track as the Principal navigates among SPs.

The opaque handle mechanism allows the Service Provider to know which of their own customers, with local accounts, has navigated to the site. It facilitates identity federation between the Principal's accounts at the Identity Provider and the Service Provider *without* transferring any PII about the Principal to the Service Provider prior to identity federation.

- ***Anonymous Identity Protocols*** – The Liberty Specifications contain protocols for sharing personalization data with a Service Provider on an anonymous basis to allow for personalization of websites and services without disclosure of the identity of the Principal or requiring the Principal to have an account with the Service Provider. Using this tool, the Principal's actual identity is not released to the Service Provider. Rather, a transient name identifier is given to the Service Provider for each session that the Identity Provider can map to its account for the Principal.

In order for a company to effectively set up a privacy framework and implement fair information practices, Liberty recognizes that there are several points within the Liberty infrastructure where privacy concerns can be addressed and enforced. Liberty calls these "Policy Enforcement Points." For example,

1. The Identity Provider can decide whether or not to authenticate a Principal based on the credentials provided and the Identity Providers authentication policy.
2. At the point where a Service Provider receives an authentication from an Identity Provider, the Service Provider can decide whether a Principal's authentication context is sufficient based on the Service Provider's authentication policy.
3. At the point where an attribute requester requests Attribute Provider information for certain attributes from a Discovery Service, the Discovery Service can decide whether to facilitate an attribute requester's interaction with an Attribute Provider, based on a Principal-managed policy and the Discovery Service's policy.
4. At the point where an Attribute Provider has received the attribute request, the Attribute Provider may mediate an attribute requester's access to its services and data, based on a Principal-managed policy and the Attribute Provider's policy.

As noted previously, Principals should have the capability to specify policies governing access to their attributes. These Principal-specific policies could be defined by assigning some predefined rules to a particular class of attribute requesters, or via a user interface enabling Principals to define sophisticated rules. Also, the Service Provider, Identity Provider, Attribute Provider, and Discovery Service will need to write policies governing their infrastructure and their service. In any event, Principal specific permission rules should override default rules if they intersect. Most of these policies can be enforced at the Policy Enforcement Points numbers 3 and 4 identified above.

The access controls noted above can be used by an implementing company to set up an access management system to determine when and under what conditions various requesters may have access to requested information. The access decisions may be determined based on (i) what information is requested, (ii) what the requester wants to do with that information, (iii) the characteristics of the requester, and/or (iv) whether the owner of the requested information (i.e., the Principal) is online with a certain authentication context, etc. An implementing company should comply with its fair information practices when setting up its access management system.

In addition, by using the usage directives, implementing companies should give Principals the opportunity to specify how their personal information will be disclosed or accessed in a default context. In addition, under the Liberty Specifications, Principals can create the opportunity to validate, override, or update the Principal's default policies at request time by specifying obligations in their policies that are fulfilled by returning *permission exceptions* to requesting attribute requesters. These permission exceptions signal that the Principal must interactively validate, override, update or supply her instructions in the context of a particular transaction.

7. Security

Security is a critical component of any computing system, providing the necessary safeguards for data protection and integrity. Security is also a fundamental basis of consumer trust and confidence in any computing environment. As with privacy, it is incumbent upon all participants to provide and practice good security. Liberty-Enabled Providers must establish and maintain the level of security appropriate to the transaction and the data. Principals should be equally vigilant to protect themselves from security risks inherent in the architecture of the Internet.

Social vulnerabilities abound when deploying Internet technology; several factors contribute. A non-exhaustive list includes: human nature, rogues, buggy software, insecure systems, bad designs, poor human interfaces, and a large installed computing base of untrusted systems. These factors present a spectrum of opportunities for malicious behavior. Robust security precautions and implementations help to limit such opportunities. The Liberty Alliance specifications are based on existing, well known Internet protocols and mechanisms. Security weaknesses inherent in the architecture of the Internet are neither increased nor eliminated by the Liberty Specifications.

Security builds trust between the user and system and allows the user to let the system perform certain actions – therefore security will actually play a significant role in every system and environment. In a perfect situation, security is built as an almost invisible but strong service, which will protect the user against different attacks and/or minimize possible negative consequences.

Strong security is an essential part of a well-working network service solution, and Liberty offers security measures and guidance designed to prevent attackers from causing troubles for Liberty-Enabled Providers or for the Principal. The Liberty Alliance has referenced both the work of the OECD and the U.S. Federal Trade Commission (FTC) in providing this security framework.

The U.S. Federal Trade Commission examined security issues in 2000. The Final Report of the FTC Advisory Committee on Online Access and Security bears quoting in part:

> Most consumers – and most companies – would expect commercial Web sites that collect and hold personal data to provide some kind of security for that data. Identifying the most effective and efficient solution for data security is a difficult task. Security is application-specific and process-specific. Different types of data warrant different levels of protection.
>
> Security – and the resulting protection for personal data – can be set at almost any level depending on the costs one is willing to incur, not only in dollars but in inconvenience for users and administrators of the system. Security is contextual: to achieve appropriate security, security professionals typically vary the level of protection based on the value of the information on the systems, the cost of particular security measures and the costs of a security failure in terms of both liability and public confidence.
>
> To complicate matters, both computer systems and methods of violating computer security are evolving at a rapid clip, with the result that computer security is more a process than a state. Security that was adequate yesterday is inadequate today. Anyone who sets detailed computer security standards – whether for a company, an industry, or a government body – must be prepared to revisit and revise those standards on a constant basis.
>
> When companies address this problem, they should develop a program that is a continuous life cycle designed to meet the needs of the particular organization or industry. The cycle should begin with an assessment of risk; the establishment and implementation of a security architecture and management of policies and procedures based on the identified risk; training programs; regular audits and continuous monitoring; and periodic reassessment of risk. These essential elements can be designed to meet the unique requirements of organizations regardless of size.[10]

In addition to the above FTC recommendations, the Liberty Alliance offers the OECD Security principles for consideration by implementing companies in accordance with particular service offerings in various jurisdictions. These nine principles are applicable to all Liberty-Enabled Providers, or in OECD terms, "participants." The specific responsibilities of any Liberty-Enabled Provider will vary according to their roles. The OECD Security Principles are:[11]

1) Awareness

Participants should be aware of the need for security of information systems and networks and what they can do to enhance security.

[10] US Federal Trade Commission Advisory Committee, "Final Report on Online Access and Security."

[11] OECD, "OECD Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security."

Awareness of the risks and available safeguards is the first line of defense for the security of information systems and networks. Information systems and networks can be affected by both internal and external risks. Participants should understand that security failures may significantly harm systems and networks under their control. They should also be aware of the potential harm to others arising from interconnectivity and interdependency. Participants should be aware of the configuration of, and available updates for, their system, its place within networks, good practices that they can implement to enhance security, and the needs of other participants.

2) Responsibility

All participants are responsible for the security of information systems and networks.

Participants depend upon interconnected local and global information systems and networks and should understand their responsibility for the security of those information systems and networks. They should be accountable in a manner appropriate to their individual roles. Participants should review their own policies, practices, measures, and procedures regularly and assess whether these are appropriate to their environment. Those who develop, design and supply products and services should address system and network security and distribute appropriate information including updates in a timely manner so that users are better able to understand the security functionality of products and services and their responsibilities related to security.

3) Response

Participants should act in a timely and co-operative manner to prevent, detect and respond to security incidents.

Recognizing the interconnectivity of information systems and networks and the potential for rapid and widespread damage, participants should act in a timely and co-operative manner to address security incidents. They should share information about threats and vulnerabilities, as appropriate, and implement procedures for rapid and effective co-operation to prevent, detect and respond to security incidents. Where permissible, this may involve cross-border information sharing and co-operation.

4) Ethics

Participants should respect the legitimate interests of others.

Given the pervasiveness of information systems and networks in our societies, participants need to recognize that their action or inaction may harm others. Ethical conduct is therefore crucial and participants should strive to develop and adopt best practices and to promote conduct that recognizes security needs and respects the legitimate interests of others.

5) Democracy

The security of information systems and networks should be compatible with essential values of a democratic society.

Security should be implemented in a manner consistent with the values recognized by democratic societies including the freedom to exchange thoughts and ideas, the free flow of information, the confidentiality of information and communication, the appropriate protection of personal information, openness and transparency.

6) Risk assessment

Participants should conduct risk assessments.

Risk assessment identifies threats and vulnerabilities and should be sufficiently broad-based to encompass key internal and external factors, such as technology, physical and human factors, policies and third-party services with security implications. Risk assessment will allow determination of the acceptable level of risk and assist the selection of appropriate controls to manage the risk of potential harm to information systems and networks in light of the nature and importance of the information to be protected. Because of the growing interconnectivity of information systems, risk assessment should include consideration of the potential harm that may originate from others or be caused to others.

7) Security design and implementation

Participants should incorporate security as an essential element of information systems and networks.

Systems, networks and policies need to be properly designed, implemented and co-ordinated to optimize security. A major, but not exclusive, focus of this effort is the design and adoption of appropriate safeguards and solutions to avoid or limit potential harm from identified threats and vulnerabilities. Both technical and non-teclmical safeguards and solutions are required and should be proportionate to the value of the information on the organization's systems and networks. Security should be a fundamental element of all products, services, systems and networks, and an integral part of system design and architecture. For end users, security design and implementation consists largely of selecting and configuring products and services for their system.

8) Security management

Participants should adopt a comprehensive approach to security management.

Security management should be based on risk assessment and should be dynamic, encompassing all levels of participants' activities and all aspects of their operations. It should include forward-looking responses to emerging threats and address prevention, detection and response to incidents, systems recovery, ongoing maintenance, review and audit. Information system and network security policies, practices, measures and procedures should be co-ordinated and integrated to create a coherent system of security. The requirements of security management depend upon the level of involvement, the role of the participant, the risk involved and system requirements.

9) Reassessment

Participants should review and reassess the security of information systems and networks, and make appropriate modifications to security policies, practices, measures and procedures.

New and changing threats and vulnerabilities are continuously discovered. Participants should continually review, reassess and modify all aspects of security to deal with these evolving risks.

8. Internet Security Vulnerabilities and Precautions

Regardless of the fair information principles and privacy practices adopted, security remains a universal vital tenet of fair information practices. Security vulnerabilities exist due to both system defects and human error or malice. There is a risk that these vulnerabilities will be exploited in an attack of some form. Attacks have different properties depending on which vulnerabilities are being exploited. Below are descriptions of the most common types of attacks.

- **Denial-of-service.** Prevents authorized users from accessing the system resource or delays authorized operations and functions. This can cause severe problems, e.g., for the Liberty-enabled providers' business and brand.
- **Dictionary.** An attack that uses a technique of successively trying all the words in some large, exhaustive list. This is a lesser kind of brute-force attack generally targeted at password authentication mechanisms. This kind of attack may lead, for example, to the impersonation of a user if an attacker can break the user's password.
- **Brute-force.** An attack that uses a technique of successively trying all possible combinations. This kind of attack may also lead, for example, to impersonation of the user if an attack can break the user's password. This attack requires more resources from an attacker compared to a dictionary attack but will provide better results when there is not a good dictionary available or when passwords are protected against being recognizable words.
- **Replay.** An attack in which a valid data transmission is maliciously or fraudulently repeated, either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a spoofing attack.
- **Spoofing.** An attack in which one system entity illegitimately poses as (assumes the identity of) another entity. There is no way to prevent an attacker from erecting a false facade that mimics the appearance and

behavior of a legitimate website. If an attacker can lure the user into visiting such a site, then the user may be fooled into believing he/she is visiting the authentic website. If the fake site happens to imitate a site at which the user typically logs in, then the fake website may be able to collect the users credentials, such as, an account name and password. The attacker could then use this information to impersonate the user at a legitimate website. There is nothing about this attack, or the weaknesses exploited to carry it out, that are specific to Liberty Specifications. It is a general vulnerability that both Principals and Liberty-Enabled Providers must guard against.

Our purpose in presenting potential routes of attack is to explain the security vulnerabilities inherent in the Internet, so that Liberty-Enabled Providers are aware of the various risks and threats and thus are able to safeguard against such risks and threats. Implementers should closely monitor security risks noted in the industry, because the situation changes rapidly, with new bugs found and existing ones corrected. Regardless of which avenue of attack is exploited, several common security weaknesses can exacerbate the potential for breach. The following sections discuss some of the more well known Internet insecurities and recommends precautions that may be taken.

8.1. Common Weaknesses

Weak Passwords – So-called "reusable passwords" are a typical means of authenticating users. Reusable means that the password is constant and used multiple times to gain access to an account. User may choose weak, "guessable" passwords which render their account susceptible to relatively simple guessing attacks (also known as "dictionary attacks"). The risks of weak passwords are significantly compounded when users choose the same password to access different accounts. This is especially true in a single sign-on environment.

Using arbitrary, unchecked, reusable passwords in conjunction with a single sign-on environment means that multiple accounts may be compromised at once by guessing one password. If the user were to choose different, unrelated passwords for each site, then the multiple sites would be better protected. But if the user does not, then the vulnerability of the multiple sites is essentially the same with or without the single sign-on environment.

Where appropriate, Identity Providers should support other forms of authentication in addition to User ID and password. In addition, Identity Providers should inform Principals how to formulate and protect their passwords from unauthorized use. In the case of password usage, IdPs can also use password-checking mechanisms to check the Principal's password. However Liberty Specifications do not force these functionalities and their usage is dependent on the IdP.

Embedded Login Forms – The ID-FF Architecture Overview 1.2 describes a deployment scenario where an Identity Provider's login form is embedded within a page presented by a Service Provider. Users often prefer the seamlessness of this embedded form mechanism, which submits the users' credentials back to the Identity Provider. However, embedded forms may permit the inadvertent exposure of Identity Provider credentials to the Service Provider in unencrypted clear text. Thus, when using authentication via embedded form, deployers should have contracts in place requiring the protection of embedded login forms.

Publicly Available Terminals– If a Principal accesses a Liberty-Enabled site using a public browser (such as at an airport kiosk or Internet cafe), there may be no rapid way for the user to terminate the session. If a Principal leaves a public browser without fully terminating the session, a subsequent Internet user may have access to the Principal's browser session.

To prevent session hijacking at a public browser, short-session times are recommended for Identity Providers. This, of course, creates a problem for Principals who leave a browser session inactive, but intend to return after some time period. This is a classic tradeoff problem, and the Liberty Alliance recommendation is in favor of security.

In addition, when during a slightly shorter interval the account shows activity by the Principal at a Service Provider, the Service Provider with whom the activity is occurring sends a refresh message to the Identity Provider. One plausible way to perform the "refresh" is for the Service Provider to send an Authentication Request (AuthnRequest) message to the Identity Provider over the preferred channel containing some defined combination of the AuthnRequest parameters, thus signaling that "the user is still active over here."

We recommend that Identity Providers have a mechanism that enables the Principal to later return using a different browser session and terminate the previous session. We also recommend that a change password feature be available

which challenges the user for their old password before accepting the new one. In addition, the Liberty Alliance currently intends to provide a refresh message mechanism in future versions of the Liberty Specifications.

Weak Cryptography – One of the most vexing issues in securing web services is that the currently installed browser base includes many browsers that only have weak 40-bit cryptography enabled. It is well known that 40-bit ciphers are considered weak and can be compromised with minimal computing effort. This poses a risk since users' encrypted communication may be easily recovered to the unencrypted form. However, the Liberty Specifications recommend cipher suites that minimally have effective secret key sizes of 112-bits. In case of signatures and public key cryptosystems the recommended minimum key length is 1024-bits.

8.2. Browser Vulnerabilities

Social vulnerabilities abound when deploying Internet technology; many factors contribute. A non-exhaustive list includes: human nature, rogues, buggy software, insecure systems, bad designs, poor human interfaces, and a large installed computing base of untrusted systems. These factors present a spectrum of opportunities to exploit one or a combination of weaknesses.

When using Internet browsers, we also need to consider their potential weaknesses. It is widely known that browsers have security-related weaknesses which can lead to information leakage. This best practices document provides information about such vulnerabilities and tips on how to avoid the most common vulnerabilities. Specifically, the security considerations section of the Liberty ID-FF Bindings and Profiles Specification describes potential vulnerabilities that are present as a consequence of implementation decisions and gives guidance in constructing name identifiers, which are a privacy-enhancing mechanism.[12] For more information about the secure implementation of Liberty version 2, please refer to the Liberty Alliance.[13] This section discusses specific browser weakness and suggests mitigation strategies. In the next section we examine security issues related to well-established protocols.

Account Federation – The Liberty Specifications enable the Principal to federate identities, binding accounts together. If done improperly, the binding could release PII. The Liberty Specifications avoid this problem by recommending implementations that generate opaque handles using arbitrary sequences of characters that map into the account. The Liberty Specifications provide for the exchange of opaque handles to federate accounts. Additionally, Identity Providers are required to create unique opaque handles for each of the Principal's federated accounts. This diminishes the threat of collusion and tracking.

Cookie Exposure – Many web browsers implement a technology known as HTTP cookies. The intended function of cookies is to supplement web protocols with state management (or session) capabilities. Cookies can be transient (used just for the lifetime of the browser session) or persistent. A persistent cookie is saved to permanent storage so that it is available the next time the user starts a web browser. The various manners in which cookies are used may sometimes violate users' privacy. For example, a cookie may collect PII without a user's consent. In addition, web browsers and other Internet software have been shown to be susceptible to inadvertent disclosure of cookies to unauthorized parties. Since a cookie may contain PII, or could even be used to impersonate a Principal, this represents an additional security and privacy risk.

The Liberty Specifications do not mandate the use of cookies, but allows for an optional cookie-based mechanism which is used to simplify single sign-on. The information in this cookie is not a privacy risk since the only information revealed are the locations or websites at which the Principal authenticates. This particular cookie is referred to as the "common domain cookie" in the Liberty Specifications.

However, in addition to the common domain cookie, it is recognized that many websites, in order to provide a "seamless" user experience, will rely on the state management properties of cookies. Note that this issue is not specific to the Liberty Specifications. Any website that uses cookies for state management – with or without Liberty Specifications – is subject to the risks regarding the exposure of cookie contents. The actual risk depends on how the website constructs their cookies, the lifetime of the cookie, and the cookie contents. If a cookie were to present the above mentioned risks, an attacker would need to discover a software defect or have access to a Principal's computer

[12] John Kemp and Tom Wason, "ID-FF Bindings & Profiles Specification."

[13] Jonathan Tourzan, "ID-WSF Architecture Overview." , ID-WSF Security and Privacy Guidelines and ID-WSF Security Profiles.

in order to exploit the vulnerability. Principals should normally have the opportunity and guidance necessary to decline cookies. For service offerings that are cookie dependent, care should be taken that the cookie does not collect or store PII.

Cross Site Scripting – Cross Site Scripting (CSS) was originally published as a CERT advisory [CSS] in February 2000. To date this is still a very common threat and has been used to trick browsers to make incorrect trust decisions such as erroneously trusting malicious code.

A CSS vulnerability could potentially be used to collect HTTP cookies or the URL history and disseminate the data to an unauthorized party. Note that this is not an issue specific to the Liberty Specifications. Combining a CSS vulnerability with a social vulnerability could potentially fully compromise a user's accounts in a single sign-on environment. The CSS vulnerability is related to browser security and avoiding this problem requires changes in browsers' architecture, which is beyond the scope of the Liberty Alliance. However, Principals should be warned as to the potential CSS vulnerability and take necessary precautions, including being very selective and limiting their use of hyperlinks in emails or instant messages to only those communications from known or trusted senders.

Related Documents/RDF – The "Related Documents Feature" (RDF) implemented in both Netscape and Internet Explorer, "Smart Browsing/What's Related" and "Show Related Documents" respectively, is known to be a very leaky channel. Essentially when the feature is enabled, the browser reports to the RDF service the referring URL and the URL to which the browser is being navigated. Like CSS, this vulnerability is inherent in the architecture of the browser. Principals should be warned of the potential for leakage and may wish to avoid the use of RDF in some circumstances.

8.3. Protocol Vulnerabilities

The Liberty Specifications were built on existing Internet technologies, meaning both browsers and protocols. The previous section discussed browser weaknesses. We now turn to protocol weaknesses. The core Internet protocols used whenever online include the Transmission Control Protocol (TCP), the Internet Protocol version 4 (IP), the User Datagram Protocol (UDP) and the Domain Name System (DNS). When browsing the Web there is another layer of protocol with the Hypertext Transfer Protocol (HTTP). These protocols are insecure. They do not support fundamental security properties of integrity, confidentiality or authenticity. In the event that a website needs to securely communicate with the browser, the Transport Layer Security version 1.0 (TLS) or Secure Socket Layer version 3.0 (SSL) protocol is inserted in a layer between TCP/IP and HTTP. This combination yields the protocol scheme known as HTTPS.

The most common tool used to access Internet resources – and thus make use of the protocols mentioned above – is the web browser. Web browsers also have insecure aspects. The remainder of this section describes vulnerabilities of these protocols and browsers and the risks and threats they pose. Please note that the issues discussed herein are present whenever anyone browses the Internet – regardless of whether Liberty Specifications are being used.

DNS – A DNS server resolves the host names found in Uniform Resource Locators (URL) into a numeric Internet address. There are two well-known ways that DNS spoofing can occur, and both can result in a user connecting to a rogue site and mistakenly believing it is real.

First, there is no assurance in the protocol that replies to queries are genuine and have not been tampered with. It has been demonstrated that rogue DNS address records can contaminate the cache of an otherwise trusted resolver. The obvious threat this spoofing attack presents is that the peer host may end up connecting to the rogue site.

The second DNS vulnerability is the possibility of a compromised DNS server. If a DNS server is hijacked, then what seems to be legitimate address resolution may also result in the user connecting to a rogue website.

In both scenarios the user has no way of knowing that he/she is not communicating with the correct host. This contributes to the "spoofing" social vulnerability discussed above

To be more resilient to these sorts of attacks, deployers can utilize SSL server authentication via HTTPS. Generally the subject name in the public key certificate bears the domain name of the server, which should match the host name in the URL used for contacting the server. Thus, along with proper certificate path validation of the server domain

name from the certificate, one can verify that both that name and the host name in the URL indeed match, and are bona fide.[14]

Structural remedies for the DNS vulnerabilities are available but not widely deployed. The Domain Name System Security Extensions define extensions which integrity protect the records returned through the use of digital signatures. Also, the security extensions provide for the optional authentication of DNS protocol interactions.

HTTP – HTTP, the prevalent web protocol, makes extensive use of URLs. URLs have a syntax enabling the embedding of adjunct information. The Liberty Specification makes extensive use of this capability. At times, such embedded information may contain sensitive data. HTTP implementations must convey and consume URLs; thus the information embedded in a URL must be visible to the endpoints. Thus there are no provisions in HTTP for protecting such sensitive, URL-embedded information. Therefore, the onus is upon HTTP implementations – browsers and web servers – to avoid inadvertently disclosing this information. The Liberty Specifications recommend implementers protect the sensitive data carried in URLs. The recommended method to protect this information in transit is for the Service Provider to protect the sensitive relay state information. Since the Service Provider is both the producer and consumer of the relay state information, the Liberty Specifications do not mandate what specific cryptographic algorithms and primitives to use. The acquired data must be stored securely. URL leaks are discussed in more detail below.

URL – The Liberty Specifications may be used to convey sensitive information between parties in URLs. There are numerous methods in which referenced URLs can leak. For example, most browsers maintain a history of visited web addresses; browsers may report to the visited website the referring URL and most websites maintain logs of activity by capturing the URLs being requested as well as the referring URL. In addition, web proxy servers are deployed within intranets to facilitate passing web traffic through the corporate firewall, and proxy servers also maintain logs of the URLs requested. Finally, firewalls typically log traffic passing through them for auditing purposes. All of these retention points pose a risk if the data in the URL is sensitive and exposed to an unauthorized party.

In designing the Liberty Specifications, common-sense efforts were made to minimize the chance of disclosing the information conveyed in the URL to an unauthorized party. Liberty Specifications prescribe the following two recommendations:

First, the Liberty Specifications recommend that entry points and subsequent protocol exchanges be initiated over a secure communication transport, TLS or SSL, which implies the URLs should specify the HTTPS scheme. By following this guidance, sensitive information contained in URLs is available only at the points where it is produced, relayed (via the browser) and consumed. More simply stated, unauthorized observers cannot see the exchanged URLs and confidential information in them.

Second, the Liberty Specifications recommend that state information passed in the URL be integrity and confidentiality protected. This is a privacy enhancing measure that limits the exposure of the Principal's Service Provider activities. It also protects the Service Provider from initiating actions on behalf of the user if the state information were fabricated or tampered with.

Network Time Protocol (NTP) Weaknesses – Both the Liberty Specifications and the Security Assertion Markup Language version 1.0 specifications employ time-based mechanisms to qualify the validity of a message and the assertions they contain. This suggests that the clocks of participating systems are synchronized so that the validity periods can be accurately verified and honored. The time-based qualifiers may also be used as countermeasures against replay attack (described above). By enforcing the validity periods, we minimize the attackers window of opportunity. The smaller the time window between the generation of an assertion and its consumption, the better the security of the protocol. Therefore, if such a countermeasure is deployed, then it will be necessary to keep the clocks of the participating sites synchronized.

NTP is designed to keep the clocks of distributed systems synchronized. NTP is not a secure protocol and measures must be taken to prevent an attacker from disrupting the service. To defend against a rogue system influencing the synchronization process by broadcasting invalid time information, authentication and access controls should be used to

[14] Procedures for performing such name matching, also known as a "server identity check," are specified in C. Newman, 1999 and J. Hodges, R. Morgan, and M. Wahl, 2000.

limit potential synchronization sources. A more thorough coverage of this topic can be found in the Sun Blueprint Series.[15]

8.4. Summary

The Liberty Alliance developed a set of standards for single sign-on and federated network identity building on currently-deployed browsers. The vulnerabilities described above, from the serious ones of Cross-Site Scripting to the more arcane problems of Network Time Protocol, are problems of the underlying infrastructure. The Liberty Alliance has made every effort to provide secure standards, but the standards are built on top of insecure existing Internet protocols and, unavoidably, present potential vulnerabilities. This is not to suggest that the Liberty Specifications are insecure, but that implementations are dependent on all the underlying protocols. Thus, care should be taken in implementation and the Liberty Alliance Implementation Guidelines should be carefully followed.[16]

9. Terminology

Below please find a glossary of certain terms used throughout this document. For more information regarding Liberty terms, please see the Liberty Glossary.[17]

Access control
The act of mediating requested access to a resource based on privilege attributes of the requester and control attributes of the requested resource.

Attribute
A distinct characteristic of a Principal. A Principal's attributes are said to describe it.

Attribute class
A predefined set of attributes, such as the constituents of a Principal's name (prefix, first name, middle name, last name, and suffix). Liberty entities may standardize such classes.

Attribute Provider (AP)
The attribute provider (AP) provides ID-PP information. Sometimes called a ID-PP provider, the AP is an ID-WSF web service that hosts the ID-PP.

Authentication
The process of verifying the ability of a communication party to "talk" in the name of a Principal.

Authentication Domain (AD)
A formal community of Liberty-enabled entities that interact using a set of well-established common rules.

Authentication session
The period of time starting after A has authenticated B and until A stops trusting B's identity assertion and requires reauthentication. Also known as "session," it is the state between a successful login and a successful logout by the Principal.

Authorization
A right or a permission that is granted to a system entity to perform an action.

Credentials
Known data attesting to the truth of certain stated facts.

[15] J. Hodges, R. Morgan, and M. Wahl, 2000.

[16] Susan Landau, "Liberty Security and Privacy Overview."

[17] Tom Wason, "Liberty Glossary."

Data
Any information that a Principal provides to an Identity Provider or a service provider.

Discovery Service (DS)
An entity that has the ability to direct attribute requesters to the relevant attribute provider who provides the requested classes of attributes for the specified Principal.

Federate
To link accounts at two or more entities together.

Federated architecture
An architecture that supports multiple entities provisioning Principals among peers within the Liberty Authentication Domain.

Federation
An association comprising any number of Service Providers and Identity Providers.

Identity
The essence of an entity, often described by its characteristics.

Identity federation
Associating, connecting, or binding multiple accounts for a given Principal at various Liberty-enabled entities within an Authentication Domain.

Identity Provider (IdP)
A Liberty-enabled entity that creates, maintains, and manages identity information for Principals and provides Principal authentication to other Service Providers within an Authentication Domain. An Identity Provider may also be a Service Provider.

Liberty-Enabled Provider
As used herein, and only herein, LEP may be either an Attribute Provider (AP), Discovery Service (DS), Service provider (SP), or Identity Provider (IdP) who collects, transfers, or receives the Personally Identifiable Information (PII) of a Principal.

Permission
Privileges granted to each user with respect to what data that the user is allowed to access and what menus options or commands he or she is allowed to use.

Personally Identifiable Information (PII)
Any data that identifies or locates a particular person, consisting primarily of name, address, telephone number, e-mail address, bank accounts, or other unique identifiers such as Social Security numbers.

Principal
A Principal is an entity that can acquire a federated identity, that is capable of making decisions, and to which authenticated actions are performed on its behalf. Examples of Principals include an individual user, a group of individuals, a corporation, other legal entities, or a component of the Liberty architecture.

Privacy
Proper handling of personal information throughout its life cycle, consistent with the preferences of the data subject.

Profile
Data comprising the broad set of attributes that may be maintained for an identity, over and beyond its identifiers and the data required to authenticate under that identity. At least some of those attributes (for example, addresses, preferences, card numbers) are provided by the Principal.

Rights Expression Languages (RELs)

A machine-based language that enables communication about usage directives. RELs allow an information provider to request intended uses of information before the information is exchanged and to designate approved uses for information exchanged during a particular transaction.

Service Provider (SP)
An entity that provides services and/or goods to Principals.

Usage directives
Directives that specify the manner in which attributes can be used, stored, and disclosed.

10. References

BBB Online, Inc. and the Council of Better Business Bureaus, Inc. "A Review of Federal and State Privacy Laws." http://www.bbbonline.org/UnderstandingPrivacy/library/fed_statePrivLaws.pdf (accessed April 11, 2003).

Berman, J., and D. Mulligan. "Privacy in the Digital Age: Work in Progress." 23 Nova Law Review 2 1999. http://www.cdt.org/publications/lawreview/1999nova.shtml.

Canadian Parliament. Canadian Privacy Act. S.C. 2000, c.5 (2003).

Cantor, Scott, Kemp, John, eds. (19 July 2003). "Liberty ID-FF Protocols and Schema Specification," Version 1.2-13,Liberty Alliance Project. http://www.projectliberty.org/specs.

CDT Web Site. "Privacy Basics: Fair Information Practices." http://www.cdt.org/privacy/guide/basic/fips.html (accessed August 6, 2003).

Center for Democracy and Technology (CDT) Web Site. "Data Privacy." http://www.cdt.org/privacy (accessed August 6, 2003).

Ellison, Gary, ed. (25 July 2003) "Liberty ID-WSF Security Mechanisms," Version 1.0-17, Liberty Alliance Project. http://www.projectliberty.org/specs.

European Parliament and the Council of 12 July 2002. Directive 2002/58/EC on privacy and electronic communications. http://europa.eu.int/smartapi/cgi/sga_doc?smartapi!celexapi!prod!CELEXnumdoc&lg=en&numdoc=32002L0058&model=guichett (accessed April 11, 2003).

European Parliament and the Council of 24 October 1995. Directive 95/46/EC on data protection. http://www.privacy.org/pi/intl_orgs/ec/eudp.html (accessed April 11, 2003).

GBDe. "Personal Data Privacy Protection Guidelines." http://www.gbde.org/privacy1.html (accessed August 6, 2003).

Hi-Ethics Web Site. www.hi-ethics.org (accessed April 2003).

Hi-Ethics. "Health Internet Ethics: Ethical Principles For Offering Internet Health Services to Consumers." http://www.hi-ethics.org/Principles/index.asp (accessed April 2003).

Hodges, J., R. Morgan, and M. Wahl. "Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security." May 2000. http://www.isi.edu/in-notes/rfc2830.txt (accessed April 11, 2003).

Hodges, Jeff, ed. (October 11, 2002). "Version 1.0 Errata." Version 1.0, Liberty Alliance Project. http://www.projectliberty.org/specs/archive/v1_0/draft-liberty-version-1-errata-00.pdf.

International Security, Trust and Privacy Alliance (ISTPA). "Privacy Framework v.1.1." http://www.istpa.org/ (accessed April 11, 2003).

Kemp, John , Wason, Tom, eds. (25 July 2003). "Liberty ID-FF Bindings and Profiles Specification," Version 1.2-14, Liberty Alliance Project. http://www.projectliberty.org/specs.

Landau, Susan, ed. (24 July 2003). "Liberty ID-WSF Security and Privacy Overview," Draft version 1.0-09, Liberty Alliance Project.

1032 Liberty Alliance. "Security Bulletin." October 10, 2002,
1033 http://www.projectliberty.org/specs/archive/v1_0/security_bulletin.html.

1034 Linn, John, ed. (25 July 2003). "Liberty Trust Models Guidelines," Draft version 1.0-07, Liberty Alliance
1035 Project. http://www.projectliberty.org/specs.

1036 Madsen, Paul , ed. (25 July 2003). "Liberty Authentication Context Specification," Version 1.2-07,Liberty
1037 Alliance Project. http://www.projectliberty.org/specs.

1038 NAI. "Self-Regulatory Principles." http://www.networkadvertising.org/aboutnai_principles.asp (accessed April
1039 2003).

1040 Newman, C. "Using TLS with IMAP, POP3 and ACAP." RFC 2595, June 1999. http://www.isi.edu/in-
1041 notes/rfc2595.txt (accessed April 11, 2003).

1042 OECD. "OECD Guidelines for the Security of Information Systems and Networks: Towards a Culture of
1043 Security." http://www.oecd.org/pdf/M00033000/M00033182.pdf (accessed April 11, 2003).

1044 OPA. "Guidelines for Online Privacy Policies." http://www.privacyalliance.org/resources/ppguidelines.shtml
1045 (accessed April 2003).

1046 Organization for Economic Co-operation and Development (OECD). Web Pages regarding privacy and security
1047 concerns. http://www.oecd.org/EN/home/0,,EN-home-43-1-no-no-no-43,00.html (accessed April 2003).

1048 Privacy International, "Privacy and Human Rights: An International Survey of Privacy Laws and Developments,
1049 2002." http://www.privacyinternational.org/survey/phr2002/ (accessed April 2003).

1050 The Global Business Dialogue on Electronic Commerce (GBDe) Web Site. http://www.gbde.org/gbde2003.html
1051 (accessed August 6, 2003).

1052 The Network Advertising Initiative (NAI) Web Site. http://www.networkadvertising.org/ (accessed April 2003).

1053 The Online Privacy Alliance (OPA) Web Site. http://www.privacyalliance.org/ (accessed April 2003).

1054 U.S. Federal Trade Commission Advisory Committee. "Final Report on Online Access and Security." (2000)
1055 http://www3.ftc.gov/acoas/papers/acoasdraft1.htm (accessed April 11, 2003).

1056 USCA. 15 U.S.C.A. § 6501 (2000); 15 U.S.C.A. § 6801 et. seq. (1999); 42 U.S.C.A. §§ 1320(d) et. seq. (1996);
1057 15 U.S.C.A. § 45(a) (2000).

1058 Wason, Thomas, ed. (06 August 2003). "Liberty Architecture Glossary," Version 1.2-09, Liberty Alliance
1059 Project. http://www.projectliberty.org/specs.

1060 Wason, Thomas, ed. (25 July 2003). "Liberty ID-FF Architecture Overview," Version 1.2-03,Liberty Alliance
1061 Project. http://www.projectliberty.org/specs.

1062 Wason, Tom, ed. (14 Apr 2003). "Liberty ID-FF Implementation Guidelines," Version 1.2-08, Liberty Alliance
1063 Project. http://www.projectliberty.org/specs.

1064 Weitzel, David, ed. (August 1, 2003). "Liberty ID-WSF Implementation Guide," Version 1.0-01, Liberty Alliance
1065 Project. http://www.projectliberty.org/specs.

Exhibit A

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

INTERNATIONAL BUSINESS MACHINES CORPORATION,	)	
Plaintiff,	)	Civil Action No. 16-122-LPS
v.	)	**JURY TRIAL DEMANDED**
GROUPON, INC.,	)	
Defendant.	)	

SECOND AMENDED IDENTIFICATION OF PREVIOUSLY DISCLOSED AUTHENTICATION OBJECTIONS

Pursuant to the Court's Order, *see* D.I. 323, Plaintiff International Business Machines Corporation ("IBM") makes the following identification of previously disclosed authentication objections to documents on Defendant Groupon, Inc.'s ("Groupon") trial exhibit list. This identification is specific to IBM's previously identified authenticity objections to documents on Groupon Exhibit List and does not waive or limit IBM's other objections, including objections on other grounds, objections to other exhibits, and objections to inaccurate descriptions. IBM reserves the right to clarify, amend, modify, and supplement the information contained in these disclosures, including based on further discussions between the parties to narrow the issues before the Court.

Ex. No.	Groupon's Description	
DX-0022	Iyengar Dep. Ex. 3 - Excerpt from Spinning the Web, Yuval Fisher (1996)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of portions of a book purportedly released in 1996, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.
DX-0029	Schmidt Dep. Ex. 5 - Groupon Webpage Screenshot - All Nashville Deals & Coupons (02/12/2018)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0030	Schmidt Dep. Ex. 6 - Groupon Webpage Screenshot - Jiffy Lube - Up To 47% Off - Madison, TN (02/12/2018)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0031	Schmidt Dep. Ex. 7 - Groupon Webpage Screenshot - Groupon Goods - Toys, Electronics, Clothing & More! Save on All You Need with Groupon (02/12/2018)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has

Ex. No.	Groupon's Description	
		remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0032	Schmidt Dep. Ex. 8 - Groupon Webpage Screenshot - Deals and Coupons for Restaurants, Fitness, Travel, Shopping, Beauty, and more (02/12/2018)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0033	Schmidt Dep. Ex. 9 - Source for website - https://www.groupon.com /browse /nashville?context =local (02/12/2018)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0057	Give Me Liberty . . . (06/01/2003)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in June of 2003, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0058	HTML and CGI Unleashed (Contents of CD) (01/01/1995)	Groupon has provided no evidence of how this CD was created and maintained, and whether or when it was updated. Groupon has not evidence showing this material is a true and accurate copy of a CD allegedly

Ex. No.	Groupon's Description	
		provided with "HTML and CGI Unleashed" allegedly released in January of 1995. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0063	Novell Debuts New digitalme 'In- the-Net' Service (10/05/1999)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a press release purportedly from October of 1999, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0067	Shibboleth-Architecture DRAFT v05 (05/02/2002)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a paper purportedly from May of 2002, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. Additionally, this document is marked as a draft.
DX-0094	Domain Delphi: Retrieving Documents Online (04/01/1986)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in April of 1986, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document

Ex. No.	Groupon's Description	
		further appears to be an excerpt of a larger document.
DX-0107	The Xerox Star: A Retrospective (09/01/1989)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in September of 1989, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.
DX-0148	Liberty Technical Glossary, version 1.3 (12/14/2004)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in December of 2004, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0149	Liberty Metadata Description and Discovery Specification, version 2.0-02 (12/14/2004)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in December of 2004, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. Further, while Groupon claims this document is

Ex. No.	Groupon's Description	
		version 2.0-02, the exhibit is marked as a DRAFT.
DX-0150	Liberty ID-FF Authentication Context Specification, version 1.2 (12/14/2004)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in December of 2004, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0151	Liberty ID-FF Implementation Guidelines, version 1.2 (04/18/2004)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in April of 2004, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0152	Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML), version 1.1, OASIS Standard (09/02/2003)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in September of 2003, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0157	Advertising on Videotex: What We've Learned (1984)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated.

Ex. No.	Groupon's Description	
		Groupon has no evidence showing this document is a true and accurate copy of an article purportedly from 1984, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.
DX-0160	The Architecture of Videotex Systems (01/01/1983)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1983, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0162	CompuServe Information Manager ("CIM")	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0163	CompuServe Navigator User's Guide (01/01/1988)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a user guide purportedly from January of 1998, nor that has remained unchanged since then, as Groupon purports.

Ex. No.	Groupon's Description	
		Groupon did not pursue any such evidence during fact or expert discovery.
DX-0164	CompuServe User's Guide - Information Manager for Windows	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0165	CSNavBrochure 08-1988	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document purportedly from August of 1988, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0167	HTML & CGI – Unleashed	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0177	NAPLPS: Beyond Videotex	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and

Ex. No.	Groupon's Description	
		accurate copy, nor that has remained unchanged, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. The document appears to be an excerpt from an unidentified larger document and it is unclear whether this is an entire article.
DX-0195	Library of Congress Record for The Complete HyperCard Handbook by Danny Goodman (08/03/1987)	Though Groupon has placed a Bates number on this document, it was not produced during discovery. Moreover, Groupon has provided no evidence linking this document to the text it purportedly reflects.
DX-0196	Library of Congress Record for Danny Goodman's HyperCard's Guide (06/01/1988)	Though Groupon has placed a Bates number on this document, it was not produced during discovery. Moreover, Groupon has provided no evidence linking this document to the text it purportedly reflects. And as IBM previously showed, the dates are in conflict. *See* D.I. 288 at 2-3.
DX-0197	Library of Congress Record for Hypercard Made Easy by William Sanders (07/08/1988)	Though Groupon has placed a Bates number on this document, it was not produced during discovery. Moreover, Groupon has provided no evidence linking this document to the text it purportedly reflects. And as IBM previously showed, the evidence shows there is no such link. *See* D.I. 288 at 1.
DX-0202	Weissman Dep. Ex. 27 - Amazon Source Code excerpt	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon

Ex. No.	Groupon's Description	
		did not pursue any such evidence during fact or expert discovery. The only witness with knowledge of Amazon's source code testified as follows: "I'm backing up and saying, even if you ask me this about 1995, when I did work there, I would not be able to prove to you that the source code marked all -- I think as "all files" something '95, that I couldn't prove to you that that was used on the web server." (Davis, Paul, 94:24-95:9, Nov. 16, 2017)
DX-0203	Library of Congress Copyright Office Catalog record for Unleashed	Though Groupon has placed a Bates number on this document, it was not produced during discovery. Moreover, Groupon has provided no evidence linking this document to the text it purportedly reflects.
DX-0204	Library of Congress Copyright Office Catalog record for Spinning the Web	Though Groupon has placed a Bates number on this document, it was not produced during discovery. Moreover, Groupon has provided no evidence linking this document to the text it purportedly reflects.
DX-0290	IBM v NYSE v NASDAQ Screenshot CapIQ.PNG	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery. Moreover, there are no indications of the origins of this image.

Ex. No.	Groupon's Description	
DX-0331	GRPN v GOOGL v AMZN v FB Screenshot CapIQ.PNG	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery. Moreover, there are no indications of the origins of this image.
DX-0339	KMS: A Distributed Hypermedia System for Managing Knowledge In Organizations (11/01/1987)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a paper purportedly from November of 1987, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0343	An Information System Based on Distributed Objects (10/04/1987)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in October of 1987, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. Further, Groupon's description claims a date of 10/4/1987 while the document itself lists dates of 10/4-8/1987. This document further appears to be an excerpt of a larger document.

Ex. No.	Groupon's Description	
DX-0344	Object-oriented GUI application development (01/01/1993)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1983, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0348	Design and Implementation of An Electronic Special Interest Magazine (01/01/1985)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a paper purportedly from January of 1985, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0350	The Power of NAPLPS: Beyond Videotex (01/01/1985)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0351	Weissman Dep. Ex. 17 - Apple Macintosh HyperCard User's Guide (01/01/1987)	Groupon has provided no evidence of whether or how this document was kept, or showing this document is a true and accurate copy of a text purportedly released in January of 1987.

Ex. No.	Groupon's Description	
DX-0352	The Complete HyperCard Handbook (01/01/1987)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1987, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0353	Weissman Dep. Ex. 18 - HyperCard Developer's Guide (01/01/1988)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1988, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0354	Weissman Dep. Ex. 20 - HyperCard Made Easy (01/01/1988)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1988, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0356	Weissman Dep. Ex. 23 - Designing the Star User Interface (04/01/1982)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and

Ex. No.	Groupon's Description	
		accurate copy of a paper purportedly from April of 1982, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.
DX-0357	Weissman Dep. Ex. 28 - The star user interface: an overview (01/01/1982)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a paper purportedly from January of 1982, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.
~~DX-0359~~	~~IBM, Sears shooting for '88 entry; New life for videotex (04/06/1987)~~	~~Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in April of 1987, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. Additionally, this document does not even purport to be the original format of any document.~~
DX-0360	Trintex Signs up 42 Advertising Clients; Is Hoping for Launch in Early '88, VP Says, (06/01/1987)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing

Ex. No.	Groupon's Description	
		this document is a true and accurate copy of an article purportedly released in June of 1987, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. Additionally, this document does not even purport to be the original format of any document and appears to omit at least an image.
DX-0363	Trintex to Aim On-Line Ads At Demographic Segments (06/30/1987)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in June of 1987, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. Additionally, this document does not even purport to be the original format of any document.
DX-0366	Videotex / Teletext: Principles & Practices (01/01/1985)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1985, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0367	Sears IBM Near a Deal to Sell Prodigy (05/08/1996)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated.

Ex. No.	Groupon's Description	
		Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in May of 1996, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0368	Wall Street IBM, Sears to Sell Prodigy for Less Than $200 Million (05/08/1996)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in May of 1996, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0371	The HTML Sourcebook	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy, nor that has remained unchanged, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0372	Web*-A Technology to Make Information Available on the Web (04/01/1995)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in April of 1995, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or

Ex. No.	Groupon's Description	
		expert discovery. This document further appears to be an excerpt of a larger document.
DX-0373	Excerpt of HTML and CGI Unleashed (01/01/1995)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1995, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.
DX-0374	Spinning the Web (01/01/1996)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1983, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0375	Physical Exhibit: AllFilesJune1995 - Amazon.com source code	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery. The only witness with knowledge

Ex. No.	Groupon's Description	
		of Amazon's source code testified as follows: "I'm backing up and saying, even if you ask me this about 1995, when I did work there, I would not be able to prove to you that the source code marked all -- I think as "all files" something '95, that I couldn't prove to you that that was used on the web server." (Davis, Paul, 94:24-95:9, Nov. 16, 2017)
DX-0376	Physical Exhibit: AllFilesJune1996 - Amazon.com source code	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery. The one witness who has testified about Amazon testified as follows: "Q. Okay. And you cannot say for certain whether the 'all files June 1996' source code repository actually contains the source code that was running on Amazon's website in June of 1996, correct? A. I -- no, it's not possible -- not possible for me to state that." (Davis, Paul, 85:6-85:12, Nov. 16, 2017)
DX-0380	Liberty ID-FF Architecture Overview, version 1.2-errata-v1.0	Groupon has provided no evidence of how this document was created and maintained, and

Ex. No.	Groupon's Description	
		whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a paper from the Liberty Alliance Project, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0381	Liberty ID-FF Protocols and Schema Specification, version 1.2-errata-v2.0 (12/14/2004)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in December of 2004, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0382	Liberty ID-FF Bindings and 962 Profiles Specification, version 1.2-errata-v2.0 (09/12/2004)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in September of 2004, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0383	Privacy and Security Best Practices, version 2.0 (11/12/2003)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a paper from the Liberty Alliance Project, nor that has remained unchanged since

Ex. No.	Groupon's Description	
		then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0390	Google authentication Source Code	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0394	Weissman Dep. Ex. 19 - The Complete HyperCard Handbook by Danny Goodman, Expanded 2nd Edition (01/01/1988)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in January of 1988, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0442	Excerpt of Spinning the Web (10/06/2017)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a text purportedly released in October of 2017, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.
~~DX-0444~~	~~"Prodigy makes the home computer a tool for shopping, travel, banking, education –~~	~~Groupon has provided no evidence of how this document~~

Ex. No.	Groupon's Description	
	~~and more," by Stanley L. Englebardt, THINK No. 2 (01/01/1989)~~	~~was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in January of 1989, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.~~
~~DX-0483~~	~~Prodigy-Trintex articles~~	~~Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery. Though this document was produced by IBM, it was provided to IBM by a third party in a context that provides no support for its authenticity.~~
DX-0638	http://www.json.org/	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0639	https://mustache.github.io/mustache.5.html (04/01/2014)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing

Ex. No.	Groupon's Description	
		this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0640	https://mustache.github.io/mustache.1.html (02/01/2014)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0641	https://developers.facebook.com/docs/facebook-login/permissions/v3.0#reference-public_profile (01/01/2018)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0642	https://developers.google.com/identity/sign-in/web/server-side-flow#implementing_the_one-time-code_flow (10/13/2017)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of what Groupon purports it to be, nor that has remained unchanged. Groupon did not pursue any such evidence during fact or expert discovery.
~~DX-0643~~	~~HyperCard Made Easy (1st ed.)~~	~~Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and~~

Ex. No.	Groupon's Description	
		~~accurate copy, nor that has remained unchanged, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.~~
DX-0645	Hinton Dep. Ex. 4- Liberty ID-FF Architecture Overview, version 1.2 (11/12/2003)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a paper purportedly from November of 2003, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0646	Hinton Dep. Ex. 5 - Liberty ID-FF Protocols and Schema Specification, version 1.2 (11/12/2003)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in November of 2003, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.
DX-0647	Hinton Dep. Ex. 6 - Liberty ID-FF Bindings and Profiles Specification, version 1.2 (11/12/2003)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of a document available in November of 2003, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery.

Ex. No.	Groupon's Description	
DX-0666	"A Security Architecture for Computational Grids" by Ian Foster et al. (1998)	Groupon has provided no evidence of how this document was created and maintained, and whether or when it was updated. Groupon has no evidence showing this document is a true and accurate copy of an article purportedly released in 1998, nor that has remained unchanged since then, as Groupon purports. Groupon did not pursue any such evidence during fact or expert discovery. This document further appears to be an excerpt of a larger document.

DATED: July 8, 2018

/s/ Edward Geist

OF COUNSEL:

John M. Desmarais
Karim Z. Oussayef
Laurie Stempler
Edward Geist
Robert C. Harrits
Brian D. Matty
Michael Matulewicz-Crowley
DESMARAIS LLP
230 Park Avenue
New York, NY 10169
Tel: (212) 351-3400

David E. Moore (#3983)
Bindu A. Palapura (#5370)
POTTER ANDERSON & CORROON LLP
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19801
Tel: (302) 984-6000
dmoore@potteranderson.com
bpalapura@potteranderson.com

Attorneys for Plaintiff
International Business Machines Corporation